Computer Science Logic 2016

CSL 2016, August 29 to September 1, 2016, Marseille, France

Edited by

Laurent Regnier

Jean-Marc Talbot

Editors

Laurent Regnier
Institut de Mathématiques de Marseille
Université d'Aix-Marseille
laurent.regnier@univ-amu.fr

Jean-Marc Talbot
Laboratoire d'Informatique Fondamentale
Université d'Aix-Marseille
jean-marc.talbot@univ-amu.fr

ACM Classification 1998
A.0 Conference Proceedings, C.2.4 Distributed Systems, D.2.4 Software/ Programs Verifications, D.3.1 Formal Definitions and Theory, D.3.3 Languages Constructs and Features, I.2.4 Knowledge Representations Formalisms and Methods, F Theory of Computation, F.4.1 Mathematical Logic

ISBN 978-3-95977-022-4

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/978-3-95977-022-4.

Publication date
August, 2016

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.

License
This work is licensed under a Creative Commons Attribution 3.0 Unported license (CC-BY 3.0): http://creativecommons.org/licenses/by/3.0/legalcode.

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

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

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.CSL.2016.0

LIPIcs – Leibniz International Proceedings in Informatics

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

Editorial Board

- Susanne Albers (TU München)
- Chris Hankin (Imperial College London)
- Deepak Kapur (University of New Mexico)
- Michael Mitzenmacher (Harvard University)
- Madhavan Mukund (Chennai Mathematical Institute)
- Catuscia Palamidessi (INRIA)
- Wolfgang Thomas (*Chair*, RWTH Aachen)
- Pascal Weil (CNRS and University Bordeaux)
- Reinhard Wilhelm (Saarland University)

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Preface
Laurent Regnier and Jean-Marc Talbot 0:ix–0:x

The Ackermann Award 2016
Thierry Coquand and Anuj Dawar 1:1–1:4

Invited Talks

Infinite Domain Constraint Satisfaction Problem
Libor Barto 2:1–2:1

Automated Synthesis: Going Distributed
Anca Muscholl 3:1–3:2

Analytic Calculi for Non-Classical Logics: Theory and Applications
Agata Ciabattoni 4:1–4:1

Coalgebraic Learning
Alexandra Silva 5:1–5:1

Contributed Talks

The Matrix Ring of a μ-Continuous Chomsky Algebra is μ-Continuous
Hans Leiß 6:1–6:15

Completeness for Coalgebraic Fixpoint Logic
Sebastian Enqvist, Fatemeh Seifan, and Yde Venema 7:1–7:19

AC Dependency Pairs Revisited
Akihisa Yamada, Christian Sternagel, René Thiemann, and Keiichirou Kusakari . 8:1–8:16

The Directed Homotopy Hypothesis
Jérémy Dubut, Eric Goubault, and Jean Goubault-Larrecq 9:1–9:16

Robust Linear Temporal Logic
Paulo Tabuada and Daniel Neider 10:1–10:21

Models of λ-Calculus and the Weak MSO Logic
Paweł Parys and Szymon Toruńczyk 11:1–11:12

On the Parallel Complexity of Bisimulation on Finite Systems
Moses Ganardi, Stefan Göller, and Markus Lohrey 12:1–12:17

Monadic Second Order Finite Satisfiability and Unbounded Tree-Width
Tomer Kotek, Helmut Veith, and Florian Zuleger 13:1–13:20

Dependence Logic vs. Constraint Satisfaction
Lauri Hella and Phokion G. Kolaitis 14:1–14:17

Quantified Constraint Satisfaction on Monoids
Hubie Chen and Peter Mayr 15:1–15:14

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Non-Homogenizable Classes of Finite Structures
Albert Atserias and Szymon Toruńczyk 16:1–16:16

Context-Free Graph Properties via Definable Decompositions
Michael Elberfeld 17:1–17:16

Successor-Invariant First-Order Logic on Graphs with Excluded Topological Subgraphs
Kord Eickmeyer and Ken-ichi Kawarabayashi 18:1–18:15

Definability of Cai-Fürer-Immerman Problems in Choiceless Polynomial Time
Wied Pakusa, Svenja Schalthöfer, and Erkal Selman 19:1–19:17

Descriptive Complexity of $\#AC^0$ Functions
Arnaud Durand, Anselm Haak, Juha Kontinen, and Heribert Vollmer 20:1–20:16

Extending Homotopy Type Theory with Strict Equality
Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus 21:1–21:17

The Seifert–van Kampen Theorem in Homotopy Type Theory
Kuen-Bang Hou (Favonia) and Michael Shulman 22:1–22:16

Guarded Cubical Type Theory: Path Equality for Guarded Recursion
Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi 23:1–23:17

Axioms for Modelling Cubical Type Theory in a Topos
Ian Orton and Andrew M. Pitts 24:1–24:19

Bar Recursion in Classical Realisability: Dependent Choice and Continuum Hypothesis
Jean-Louis Krivine 25:1–25:11

Extracting Non-Deterministic Concurrent Programs
Ulrich Berger 26:1–26:21

Polymorphic Game Semantics for Dynamic Binding
James Laird 27:1–27:16

High-Quality Synthesis Against Stochastic Environments
Shaull Almagor and Orna Kupferman 28:1–28:17

Hedging Bets in Markov Decision Processes
Rajeev Alur, Marco Faella, Sampath Kannan, and Nimit Singhania 29:1–29:20

Minimizing Regret in Discounted-Sum Games
Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin 30:1–30:17

Easy to Win, Hard to Master: Optimal Strategies in Parity Games with Costs
Alexander Weinert and Martin Zimmermann 31:1–31:17

A Sequent Calculus for a Modal Logic on Finite Data Trees
David Baelde, Simon Lunel, and Sylvain Schmitz 32:1–32:16

Axiomatizations for Propositional and Modal Team Logic
Martin Lück 33:1–33:18

Semantics for "Enough-Certainty" and Fitting's Embedding of Classical Logic in S4
Gergei Bana and Mitsuhiro Okada 34:1–34:17

Counting in Team Semantics
Erich Grädel and Stefan Hegselmann 35:1–35:18

The Logical Strength of Büchi's Decidability Theorem
Leszek Aleksander Kołodziejczyk, Henryk Michalewski, Cécilia Pradic, and Michał Skrzypczak 36:1–36:16

The Height of Piecewise-Testable Languages with Applications in Logical Complexity
Prateek Karandikar and Philippe Schnoebelen 37:1–37:22

One-Dimensional Logic over Words
Emanuel Kieroński 38:1–38:15

Quine's Fluted Fragment is Non-Elementary
Ian Pratt-Hartmann, Wiesław Szwast, and Lidia Tendera 39:1–39:21

Free-Cut Elimination in Linear Logic and an Application to a Feasible Arithmetic
Patrick Baillot and Anupam Das 40:1–40:18

The Relational Model Is Injective for Multiplicative Exponential Linear Logic
Daniel de Carvalho 41:1–41:19

Infinitary Proof Theory: the Multiplicative Additive Case
David Baelde, Amina Doumane, and Alexis Saurin 42:1–42:17

Preface

The annual conference Computer Science Logic (CSL 2016) of the European Association for Computer Science Logic (EACSL) was held in Marseille, from August 29 to September 1, 2016. CSL started as a series of international workshops on Computer Science Logic, and became at its sixth meeting the Annual Conference of the EACSL. This conference was the 30th workshop and 25th EACSL conference. The conference was jointly organized by the "Modelisation and Verification" Research Group of the Laboratoire d'Informatique Fondamentale de Marseille and the "Logic of programming" Research Group of the Institut de Mathématiques de Marseille.

A total of 122 abstracts were registered and 97 of these were followed by full papers submitted to CSL 2016. After a three week electronic meeting, the programme committee selected 37 papers for presentation at the conference and publication in the proceedings. Each paper was assigned to at least three programme committee members. The overall quality of the submissions was very high. This made the work of the programme committee a difficult task. There was an implicit limit on the number of accepted papers imposed by the four-days duration of the conference, hence the programme committee had to reject several good papers in the end. The programme committee was assisted by a number of external reviewers providing additional expertise. The list of external reviewers is included in these proceedings.

In addition to the contributed talks, CSL 2016 had four invited speakers: Libor Barto (Charles University Prague), Agata Ciabattoni (Technische Universität Wien), Anca Muscholl (Université de Bordeaux), and Alexandra Silva (University College London). The invited speakers have contributed an abstract which is included in the proceedings.

The Ackermann Award is the EACSL Outstanding Dissertation Award for Logic in Computer Science. This year, the jury decided to give the Ackermann Award for 2016 to Nicolai Kraus. The award was officially presented at the conference on August 31, 2016. The citation of the award, an abstract of the thesis and a biographical sketch of the recipient written by Thierry Coquand and Anuj Dawar is included in the proceedings.

Four satellite workshops were also organized before and after the conference on various related topics:

- CRECOGI: Concurrent, Resourceful and Effectful Computation by Geometry of Interaction (August 28);
- LCC'16: Logic and Computational Complexity 2016 (September 2 and 3);
- PLRR: Parametricity, Logical Relations and Realizability (September 2);
- QSLC: Quantitative Semantics of Logic and Computation (September 2 and 3).

We wish to thank all members of the programme committee and all external reviewers for their work on reviewing and discussing the papers. Our thanks also go to the members of the Organizing Committee and the Workshops Organizers for their valuable help to organize CSL 2016. Finally we want to address our warmfull thanks to Marc Herbstritt from the Dagstuhl/LIPIcs team for assisting us in the production of the proceedings.

The conference received support from the Laboratoire d'Informatique Fondamentale de Marseille, from the Institut de Mathématiques de Marseille, from Aix-Marseille University, from the ARCHIMEDE Labex (ANR-11-LABX-0033) and the A*MIDEX project (ANR-11-IDEX-0001-02) funded by the "Investissements d'Avenir" French Government program, managed by the French National Research Agency (ANR). We thank these organizations for their generous supports.

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

It is not possible to end this preface without a special notice for Helmut Veith who tragically disappeared this year. Helmut was a member of the EACSL board and has contributed twice to the organization of CSL. These proceedings contain one of his scientific publications. All the members of the programme and organization committees wish to express their sadness and sympathy to his colleagues, friends and relatives.

Conference Organization

Programme Committee

- Giovanna D'Agostino (University of Udine)
- Christel Baier (Technical University of Dresden)
- Michael Benedikt (Oxford University)
- Manuel Bodirsky (Technical University of Dresden)
- Sam Buss (University of California, San Diego)
- Luis Caires (Departamento de Informatica – Universidade Nova de Lisboa)
- Thomas Ehrhard (Laboratoire PPS – Université Paris Diderot-Paris 7)
- Emmanuel Filiot (Université Libre de Bruxelles)
- Silvio Ghilardi (Dipartimento di Matematica – Università degli Studi di Milano)
- Valentin Goranko (Stockholm University)
- Anna Ingólfsdóttir (Reykjavik University)
- Laura Kovács (Chalmers University of Technology)
- Marta Kwiatkowska (University of Oxford)
- Christof Löding (Rheinisch-Westfälische Technische Hochschule Aachen)
- Assia Mahboubi (INRIA)
- Guy McCusker (University of Bath)
- Magdalena Ortiz (Institute of Information Systems – Vienna University of Technology)
- Sophie Pinchinat (IRISA Rennes)
- Laurent Regnier (Institut de Mathématiques de Marseille – Université d'Aix-Marseille)
- Sylvain Salvati (INRIA)
- Uli Sattler (University of Manchester)
- Peter Selinger (Dalhousie University)
- Thomas Streicher (Technische Universität Darmstadt)
- Jean-Marc Talbot (Laboratoire d'Informatique Fondamentale – Université d'Aix-Marseille)
- Paweł Urzyczyn (University of Warsaw, Institute of Informatics)
- Luca Viganò (King's College London)

Organizing Committee

- Emmanuel Beffara (I2M – Université d'Aix-Marseille)
- Benjamin Monmege (LIF – Université d'Aix-Marseille)
- Laurent Regnier (I2M – Université d'Aix-Marseille)
- Pierre-Alain Reynier (LIF – Université d'Aix-Marseille)
- Luigi Santocanale (LIF – Université d'Aix-Marseille)
- Jean-Marc Talbot (LIF – Université d'Aix-Marseille)
- Lionel Vaux (I2M – Université d'Aix-Marseille)

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier

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

External Reviewers

Andreas Abel
Luca Aceto
Bahareh Afshari
Dimitri Ara
Guillaume Aucher
Marc Bagnol
Patrick Bahr
Nicolas Basset
Francisco Bavera
Emmanuel Beffara
Stefano Berardi
Ulrich Berger
Olaf Beyersdorff
Paul Blain Levy
Frédéric Blanqui
Valentin Blot
Mikołaj Bojańczyk
Eduardo Bonelli
Romain Brenguier
James Brotherston
Guillaume Brunerie
Michaël Cadilhac
Olivier Carton
Witold Charatonik
Hubie Chen
Taolue Chen
Yannick Chevalier
Cyril Cohen
Thomas Colcombet
Rodica Condurache
Thierry Coquand
Bruno Courcelle
Pierre Courtieu
Pierre-Louis Curien
Łukasz Czajka
Wojciech Czerwiński
Luc Dartois
Anupam Das
Anuj Dawar
Dario Della Monica
Henry Deyoung
Gilles Dowek
Klaus Dräger
Clemens Dubslaff
Arnaud Durand
Alessandro Facchini
Maribel Fernandez
Santiago Figueira
Eric Finster
Melvin Fitting
Vojtech Forejt
Tim French
Didier Galmiche
Nicola Gambino
Olivier Gauwin
Simon Gay
Herman Geuvers
Niels Bjørn Bugge Grathwohl
Charles Grellois
Alberto Griggio
Martin Grohe
Erich Grädel

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Ernst Moritz Hahn
Helle Hvid Hansen
Hugo Herbelin
Claudio Hermida
William Hesse
Roger Hindley
Naohiko Hoshino
Xiaowei Huang
Mikolas Janota
Ismaël Jecker
Cezary Kaliszyk
Krzysztof Kapulkin
Joachim Klein
Bartek Klin
Antonina Kolokolova
Tomer Kotek
Alexander Kurz
Antti Kuusisto
Victor Lagerkvist
James Laird
Martin Lang
Francois Laroussinie
Luca Laurenti
Stephane Le Roux
Giacomo Lenzi
Thomas Leventis
Nathan Lhote
Daniel R. Licata
Jiamou Liu
Daniel Lokshtanov
Etienne Lozes
Denis Lugiez
Peter LeFanu Lumsdaine
Anders Lundstedt
Marcello Mamino
Alberto Marcone
Barnaby Martin
Bastien Maubert
Damiano Mazza
Paul-André Melliès
Daniel Mery
George Metcalfe
Marino Miculan
Samuel Mimram
Kenji Miyamoto
Benjamin Monmege
Stefan Monnier
Georg Moser
Antoine Mottet
Moritz Mueller
Martin Mundhenk
Filip Murlak
Anders Mörtberg
David Müller
Luke Ong
Erik Palmgren
Nicola Paoletti
Matteo Pascucci
Fabio Pasquali
Vitaly Perevoshchikov
Guillermo Perez
Frank Pfenning
Carla Piazza
Thomas Place
Detlef Plump
Alberto Policriti

Tim Porter
François Pottier
Gabriele Puppis
Pierre-Marie Pédrot
Jaime Ramos
Jakob Rehof
Colin Riba
David Richerby
Simon Robillard
Wren Romano
Neil Ross
Joshua Sack
Antonino Salibra
Luigi Santocanale
Andrea Schalk
Lutz Schröder
Aleksy Schubert
François Schwarzentruber
Cristina Serban
Mahsa Shirmohammadi
Michał Skrzypczak
Sam Staton
Martin Suda
Mária Svoreňová
Kazushige Terui
Jana Tumova
Christian Urban
Viktor Vafeiadis
Matthijs Vakar
Lionel Vaux
Steen Vester
Petrucio Viana
Laurent Vigneron
Marco Volpe
Igor Walukiewicz
Piotr Witkowski
James Worrell
Marc Zeitoun
Thomas Zeume
Margherita Zorzi

The Ackermann Award 2016

Thierry Coquand*[1] and Anuj Dawar*[2]

1 **Chalmers University of Technology, Gothenburg, Sweden**
2 **University of Cambridge, Cambridge, U.K.**

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 2016 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.2016.1

Category Award Description

1 The Ackermann Award 2016

The twelfth Ackermann Award is presented at CSL'16 in Marseille, France. The 2016 Ackermann Award was open to PhD dissertations in topics specified by the 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 2014 and 31 December 2015. The Jury received fifteen nominations for the Ackermann Award 2016. The candidates came from a number of different countries across the world. The institutions at which the nominees obtained their doctorates represent nine countries in Europe, North America, the Middle East and Australia.

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 decided to award the **2016 Ackermann Award** to:

> Nicolai Kraus from Germany, for his thesis
> *Truncation Levels in Homotopy Type Theory*
> approved by the University of Notthingham, UK, in 2015,
> supervised by Thorsten Altenkirch.

Citation. Nicolai Kraus receives the *2016 Ackermann Award* of the European Association of Computer Science Logic (EACSL) for his thesis

> *Truncation Levels in Homotopy Type Theory.*

* For the Jury of the EACSL Ackermann Award.

© Thierry Coquand and Anuj Dawar;
licensed under Creative Commons License CC-BY

His thesis makes fundamental contributions to the field of Homotopy Type Theory. In particular, he resolves an important open question on the truncation level of the nth univalent universe. He also obtains deep results on propositional truncation. In obtaining his results, he develops sophisticated mathematical and logical methods which are certain to play an important role in the further development of this field.

Background of the Thesis. Dependent type theory was introduced by the mathematician N.G. de Bruijn as a formalism particularly well-adapted to represent in automated mathematical reasoning. The following quote from D. Knuth is a good illustration of its originality:

> This reminds me of the very interesting language AUTOMATH, invented by Dijkstra's colleague (and next-door neighbor) N.G. de Bruijn. AUTOMATH is not a programming language, it is a language for expressing proofs of mathematical theorems. The interesting thing is that AUTOMATH works entirely by type declarations, without any need for traditional logic! I urge you to spend a couple of days looking at AUTOMATH, since it is the epitome of the concept of type.

This formalism extends *simple type theory* (A. Church 1940), allowing the representation of arbitrary structures, and exploiting systematically the idea of "propositions-as-types". It takes as primitive the notion of a *family of types* over a given type together with the operation of *dependent product* and *dependent sum*. Recently, a proof of the 4-colour Theorem (2004), and then a proof of the Feit-Thompson Theorem (2012), were formally expressed in a formal system based on type theory, showing that this approach scales to non-trivial results. These representations however were limited to *discrete* structures (finite graphs, or finite groups), and it was not entirely clear how to represent in this formal system more general structures, not necessarily discrete, and notions essential in some parts of contemporary mathematics and computer science, such as the notion of category. Connected to this, the *identity type*, introduced by P. Martin-Löf in 1973, with an elimination rule which exploited fully the notion of dependent types, had a somewhat mysterious status. It was not clear in particular how to generalize to dependent types the *extensionality axiom*, which plays a fundamental role in set theory and in simple type theory. An answer to these questions was provided by the formulation of the *univalence axiom* (Voevodsky 2010), which can be seen as a general expression of the extensionality axiom. Voevodsky also introduced a natural stratification of types, according to the complexity of their associated identity types: first propositions (where any two objects are equal), then sets (where any equality type is a proposition), then groupoids (where any equality type is a set), 2-groupoid, *etc.* Following this description, one can view type theory as a *generalization* of set theory, where sets appears as types having a particularly simple notion of identity. Voevodsky also introduced a new modal operation on types, the *propositional truncation*. These discoveries have deeply transformed our understanding of type theory, and a special year (2012–13) was organized at the Institute of Advanced Study in Princeton to better analyse their implications. Some open problems were formulated during this meeting. One such problem was to understand if there is some connection between the intuitive notion of "size" of a type and the complexity of its equality. More precisely, Martin-Löf had introduced in type theory "larger and larger" universes U_0, U_1, $U_2, \ldots$ Is it the case that the corresponding equality type also becomes more and more complex? It is for instance easy to show from the univalence axiom that U_0 itself cannot be a set, but must at least be a groupoid. With more effort, one can show that U_1 is not a groupoid, but at least a 2-groupoid. Can one show in general that U_n is not an n-groupoid? This question is already a challenge for $n = 2$.

Kraus' Thesis. A first impressive contribution (joint with Christian Sattler) of this thesis is to present a solution to this open problem, at any level n. Furthermore, this result is completely formalized in the interactive proof system Agda. This argument relies on an elegant general principle: the *Local-Global Looping Principle*, which states that an $(n+2)$-loop in the universe with base point X is the same as a family of $(n+1)$-loops in X.

The second contribution consists in a deep analysis of the operation of propositional truncation. By definition, we know that, if the type B is a proposition, a map from the propositional truncation of a type A to the type B corresponds exactly to a map from A to B. A natural question is: what happens if B has a more complex notion of equality, for instance if B is a set? How to give then a map from the propositional truncation of A to the type B? For analysing this question, Nicolai Kraus introduces the notion of a *weakly constant* map, i.e. a map such that the images of any two points in the domain are equal in the codomain. This notion is subtle when the equality in the codomain is complex. A general result about this notion is the *Fixed Point Lemma* (also now known as *Kraus' Lemma*) which states that the type of fixed-points of a weakly constant endomap is always a proposition. Nicolai Kraus presents then a general characterization of maps from the propositional truncation of a type, without any restriction on the complexity of the codomain.

This thesis contains several other surprising results. Here we highlight two of them. The first is that one can show that the propositional truncation of the sum of two propositions has the universal property of the *join* operation (an important operation in homotopy theory). The second is that it is possible to build a type that has exactly one non-trivial homotopy group on level n, and this, *only* using the univalence axiom. This type can be seen as an approximation of an Eilenberg-Mac Lane space, also important in homotopy theory.

The thesis is coherently written, in a fluent, lucid and scholarly manner. It gives a lucid account of a number of deep, original and technically non-trivial contributions to the field of dependent type theory, answering difficult open questions and formulating new techniques and notions which will play an important role in its future development.

Biographical Sketch. Nicolai Kraus completed his early education in Bavaria, being a representative for the German Mathematics Olympiad in ever year in the period 2002–07. He pursued his undergraduate education at the Ludwig Maximilains University in Munich obtaining a Bachelor's degree in Mathematics in 2010 and a second Bachelor's in Computer Science in 2011. He then pursued a PhD at the University of Nottingham in England under the supervision of Thorsten Altenkirch. Since 2015 he is is post-doctoral Research Fellow at Nottingham.

2 Jury

The Jury for the **Ackermann Award 2016** 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:

- Thierry Coquand (Chalmers University of Gothenburg),
- 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 7),
- Simona Ronchi Della Rocca (University of Torino), the vice-president of EACSL.

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

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

Infinite Domain Constraint Satisfaction Problem

Libor Barto

Department of Algebra, Faculty of Mathematics and Physics, Charles University in Prague, Prague, Czech Republic

Abstract

The computational and descriptive complexity of finite domain fixed template constraint satisfaction problem (CSP) is a well developed topic that combines several areas in mathematics and computer science. Allowing the domain to be infinite provides a way larger playground which covers many more computational problems and requires further mathematical tools. I will talk about some of the research challenges and recent progress on them.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems/Complexity of proof procedures, F.4.1 Mathematical Logic/Logic and constraint programming

Keywords and phrases Descriptive complexity, Constraint Satisfaction Problem

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.2

Category Invited Talk

References

1 Libor Barto, Jakub Oprsal, and Michael Pinsker. The wonderland of reflections. *CoRR*, abs/1510.04521, 2015. URL: `http://arxiv.org/abs/1510.04521`.

2 Libor Barto and Michael Pinsker. The algebraic dichotomy conjecture for infinite domain constraint satisfaction problems. *CoRR*, abs/1602.04353, 2016. URL: `http://arxiv.org/abs/1602.04353`.

3 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. *CoRR*, abs/1201.0856, 2012. URL: `http://arxiv.org/abs/1201.0856`.

© Libor Barto;
licensed under Creative Commons License CC-BY

Automated Synthesis: Going Distributed

Anca Muscholl

LaBRI, CNRS, Université Bordeaux, Bordeax, France; and
Technical University of Munich, IAS, Munich, Germany

Abstract

Synthesis is particularly challenging for concurrent programs. At the same time it is a very promising approach, since concurrent programs are difficult to get right, or to analyze with traditional verification techniques. The talk provides an introduction to distributed synthesis in the setting of Mazurkiewicz traces, and its applications to decentralized runtime monitoring.

1998 ACM Subject Classification D.2.4 Software/Program Verification, D.1.3 Concurrent Programming, I.2.2 Automatic Programming

Keywords and phrases Concurrent programs, Distributed synthesis, Runtime monitoring

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.3

Category Invited Talk

References

1. Parosh Abdulla, Stavros Aronis, Bengt Jonsson, and Konstantinos Sagonas. Optimal dynamic partial order reduction. In *POPL'14*, pages 373–384. ACM, 2014.
2. S. Akshay, Ionut Dinca, Blaise Genest, and Alin Stefanescu. Implementing realistic asynchronous automata. In *FSTTCS'13*, LIPIcs, pages 213–224. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013.
3. Rajeev Alur, Ken McMillan, and Doron Peled. Model-checking of correctness conditions for concurrent objects. In *LICS'96*, pages 219–228. IEEE, 1996.
4. Pavol Cerný, Thomas A. Henzinger, Arjun Radhakrishna, Leonid Ryzhyk, and Thorsten Tarrach. Efficient synthesis for concurrency by semantics-preserving transformations. In *CAV'13*, LNCS, pages 951–967. Springer, 2013.
5. Alonzo Church. Logic, arithmetics, and automata. *Proceedings of the International Congress of Mathematicians*, 1962.
6. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Verlag, 1981.
7. R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Information and Computation*, 106:159–202, 1993.
8. Philippe Darondeau and Laurie Ricker. Distributed control of discrete-event systems: A first step. *Transactions on Petri Nets and Other Models of Concurrency*, 6:24–45, 2012.
9. Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.
10. Azadeh Farzan and Parthasarathy Madhusudan. Monitoring atomicity in concurrent programs. In *CAV'08*, LNCS, pages 52–65. Springer, 2008.
11. Bernd Finkbeiner and Ernst-Ruediger Olderog. Petri games: Synthesis of distributed systems with causal memory. In *GandALF'14*, EPTCS, pages 217–230, 2014.
12. Bernd Finkbeiner and Sven Schewe. Uniform distributed synthesis. In *LICS'05*, pages 321–330. IEEE, 2005.

© Anca Muscholl;
licensed under Creative Commons License CC-BY

13 Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In *FSTTCS'04*, LNCS, pages 275–286. Springer, 2004.

14 Paul Gastin and Nathalie Sznajder. Fair synthesis for asynchronous distributed systems. *ACM Transactions on Computational Logic*, 14(2):9, 2013.

15 Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz. Optimal Zielonka-type construction of deterministic asynchronous automata. In *ICALP'10*, LNCS, pages 52–63. Springer, 2010.

16 R.M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3:147–195, 1969.

17 Siddharth Krishna and Anca Muscholl. A quadratic construction for Zielonka automata with acyclic communication structure. *Theoretical Computer Science*, 503:109–114, 2013.

18 Orna Kupferman and Moshe Y. Vardi. Synthesizing distributed systems. In *LICS'01*, pages 389–398. IEEE, 2001.

19 Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Operating Systems*, 21(7):558–565, 1978.

20 P. Madhusudan, P. S. Thiagarajan, and S. Yang. The MSO theory of connectedly communicating processes. In *FSTTCS'05*, LNCS, pages 201–212. Springer, 2005.

21 P. Madhusudan and P.S. Thiagarajan. Distributed control and synthesis for local specifications. In *ICALP'01*, volume 2076 of *LNCS*, pages 396–407. Springer, 2001.

22 Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

23 Madhavan Mukund and Milind A. Sohoni. Keeping track of the latest gossip in a distributed system. *Distributed Computing*, 10(3):137–148, 1997.

24 Anca Muscholl and Igor Walukiewicz. Distributed synthesis for acyclic architectures. In *FSTTCS'14*, LIPIcs, pages 639–651. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014.

25 Doron A. Peled. All from one, one for all: on model checking using representatives. In *CAV'93*, LNCS, pages 409–423. Springer, 1993.

26 G. L. Peterson and J. H. Reif. Multi-person alternation. In *FOCS'79*, pages 348–363. IEEE, 1979.

27 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. ACM POPL*, pages 179–190, 1989.

28 Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *FOCS'90*. IEEE, 1990.

29 M. O. Rabin. *Automata on Infinite Objects and Church's Problem*. American Mathematical Society, Providence, RI, 1972.

30 P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(2):81–98, 1989.

31 Koushik Sen, Abhay Vardhan, Gul Agha, and Grigore Rosu. Decentralized runtime analysis of multithreaded applications. In *International Parallel and Distributed Processing Symposium (IPDPS 2006)*. IEEE, 2006.

32 Alin Stefanescu. *Automatic synthesis of distributed transition systems*. PhD thesis, Universität Stuttgart, 2006.

33 W. Zielonka. Notes on finite asynchronous automata. *RAIRO–Theoretical Informatics and Applications*, 21:99–135, 1987.

34 W. Zielonka. Asynchronous automata. In G. Rozenberg and V. Diekert, editors, *Book of Traces*, pages 175–217. World Scientific, Singapore, 1995.

Analytic Calculi for Non-Classical Logics: Theory and Applications

Agata Ciabattoni

Institute of Computer Languages, Technische Universität Wien, Vienna, Austria

Abstract

The possession of a suitable proof-calculus is the starting point for many investigations into a logic, including decidability and complexity, computational interpretations and automated theorem proving. By suitable proof-calculus we mean a calculus whose proofs exhibit some notion of subformula property ('analyticity'). In this talk we describe a method for the algorithmic introduction of analytic sequent-style calculi for a wide range of non-classical logics starting from Hilbert systems. To demonstrate the widespread applicability of this method, we discuss how to use the introduced calculi for proving various results ranging from Curry-Howard isomorphism to new interpretative tools for Indology.

1998 ACM Subject Classification F.4.1 Mathematical Logic/Proof theory

Keywords and phrases Proof theory, Fuzzy logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.4

Category Invited Talk

References

1 Federico Aschieri, Agata Ciabattoni, and Francesco A. Genco. Curry-Howard Correspondance for Goedel Logic: from Natural Deduction to Parallel Computation. Submitted.

2 Agata Ciabattoni, Elisa Freschi, Francesco A. Genco, and Björn Lellmann. Mīmāmsā deontic logic: Proof theory and applications. In *Automated Reasoning with Analytic Tableaux and Related Methods – 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015. Proceedings*, pages 323–338, 2015.

3 Agata Ciabattoni, Nikolaos Galatos, and Kazushige Terui. From axioms to analytic rules in nonclassical logics. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 229–240, 2008.

4 Agata Ciabattoni and Revantha Ramanayake. Power and limits of structural display rules. *ACM TOCL*, 17(3), 2016.

© Agata Ciabattoni;
licensed under Creative Commons License CC-BY

Coalgebraic Learning

Alexandra Silva

Programming Principles, Logic and Verification Group, University College London, London, U.K.

Abstract

The area of automata learning was pioneered by Angluin in the 80's [1]. Her original algorithm, which applied to regular languages and deterministic automata, has been extended to various types of automata and used in software and hardware verification. In this talk, we will take an abstract perspective at automata learning. We show how the correctness of the original algorithm and many extensions can be captured in one proof using coalgebraic techniques. We also show that a novel algorithm for nominal automata can be derived from the abstract framework.

1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages

Keywords and phrases Automata learning, coalgebraic techniques

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.5

Category Invited Talk

References

1 Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. doi:10.1016/0890-5401(87)90052-6.

© Alexandra Silva;
licensed under Creative Commons License CC-BY

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier; Article No. 5; pp. 5:1–5:1

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

The Matrix Ring of a μ-Continuous Chomsky Algebra is μ-Continuous

Hans Leiss

Centrum für Informations- und Sprachverarbeitung,
Ludwig-Maximilians-Universität München,
Oettingenstr. 67, D-80538 München, Germany
leiss@cis.uni-muenchen.de

Abstract

In the course of providing an (infinitary) axiomatization of the equational theory of the class of context-free languages, Grathwohl, Kozen and Henglein (2013) have introduced the class of μ-continuous Chomsky algebras. These are idempotent semirings where least solutions for systems of polynomial inequations (i.e. context-free grammars) can be computed iteratively and where multiplication is continuous with respect to the least fixed point operator μ. We prove that the matrix ring of a μ-continuous Chomsky algebra also is a μ-continuous Chomsky algebra.

1998 ACM Subject Classification F 4.3 Formal languages, F 4.2 Grammars and Other Rewriting Systems, D 3.3 Language Constructs, F.3.3 Studies of Program Constructs

Keywords and phrases context-free language, fixed point operator, idempotent semiring, matrix ring, Chomsky algebra

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.6

1 Introduction

The set of context-free languages over an alphabet X has been characterized algebraically by Gruska [4] as the closure of the finite languages over X under (binary) union $+$, elementwise concatenation $\cdot$, and a least fixed point operator μ. The natural definition of a context-free language is by a simultaneous definition involving auxiliary languages, which suggests using an n-ary fixed point operator to denote the solution of a system $x_i \geq p_i(x_1, \ldots, x_n)$, $1 \leq i \leq n$, of polynomial inequations. Bekić [1], deBakker and Scott [3] noticed that a unary least fixed point operator suffices to name the components of an n-ary least fixed point, provided (i) every countable ascending chain has a supremum in the underlying partial order, and (ii) the functions p_i in the systems $x_i \geq p_i$ are componentwise continuous, i.e. map the sup of an ascending chain to the sup of the image of the chain.

Terms involving a unary fixed point operator μ in addition to semiring operations $+$, $\cdot$, 0, 1, here called μ-terms, are therefore a means to name the context-free languages. Kleene's iteration operator * amounts to a special case of recursion μ, namely to head- or tail recursion, $r^* = \mu x(rx + 1)$. Small fragments of the equational theory of context-free languages using μ-terms had been studied in [9, 10], but only recently a complete (infinitary) axiomatization has been given by Grathwohl, Henglein and Kozen [11]. Essentially, it says that μxt is the sup of all finite iterations of t and that the semiring operations are continuous with respect to definable increasing chains. In the models of this theory, the μ-continuous Chomsky algebras of [11], the partial order derived from $+$ need *not* be complete.

Simple equations between μ-terms relate head- to tail recursion and reflexive transitive closure [9], and most of the context-free grammar normalization algorithms can be derived as equations between μ-terms from minimality assumptions on μ and the semiring properties

© Hans Leiß;
licensed under Creative Commons License CC-BY

of $+$, $\cdot$, 0, 1 [10]. In particular, one can express the elimination of a head recursion by a tail recursion as an equation between μ-terms (assuming x is not an initial factor in s)

$$\mu x(xr + s) = \mu x \mu y(yr + s) = \mu x(s \cdot r^*) = \mu x(s \cdot \mu y(ry + 1)), \qquad \text{(with fresh } y\text{)}$$

and then obtain (an efficient algorithm for) the transformation of context-free grammars to Greibach normal form as the matrix version of this equation; an example is given in [10].

It is therefore of some interest to know if a property involving the unary fixed point operator lifts to the n-ary or the matrix (n^2-ary) case. A related question, which we consider here, is whether the matrix algebra of an idempotent semiring that is closed under least fixed points is itself closed under least fixed points.

We will reduce the matrix case to the vector case, and the vector case to the unary one by using the Bekić-Scott equations. However, the algebras we are considering, like the algebra of all context-free languages over X, are *not* closed under countable unions of ascending chains. Therefore, we must take some care to show that all the least fixed points involved in Bekić's reduction exist. We separate arguments about existence of least fixed points from the question of iteratively computing them, restricting the μ-continuity condition of Grathwohl e.a.[11] to the second question. Thus, if M is a Chomsky algebra, i.e. all polynomial inequation systems have least solutions, so is $Mat_{n,n}(M)$, and if M is a μ-continuous Chomsky algebra, $Mat_{n,n}(M)$ also is. In particular, the μ-continuity condition gives rise to its own matrix version. For several conditions on Kleene's iteration * in regular algebra, Conway [2] had noted that the unary case implies the matrix case; likewise, Kozen's axioms for * imply their own matrix versions [8].

2 Park μ-Semirings

▶ **Definition 1.** A **semiring** $(M, +, \cdot, 0, 1)$ is a set M with binary operations $+, \cdot$ on M and elements $0, 1 \in M$ such that $+$ is associative and commutative with neutral element 0, $\cdot$ is associative with neutral element 1 and annihilator 0, and $\cdot$ distributes over $+$ from both sides. An **idempotent semiring** is a semiring where $+$ is idempotent.

▶ **Definition 2.** Let X be a set of variables. The set of **μ-terms over X** is defined by the grammar

$$t := x \mid 0 \mid 1 \mid (s \cdot t) \mid (s + t) \mid \mu x t.$$

A term not containing μ will be called **algebraic** or (somewhat unprecise) a **polynomial**. The free occurrences of variables in a term are defined as usual. By $\mathit{free}(t)$ we denote the set of variables having a free occurrence in t; in particular, $\mathit{free}(\mu x t) = \mathit{free}(t) \setminus \{x\}$. By $t(x_1, \ldots, x_n)$ we indicate $\mathit{free}(t) \subseteq \{x_1, \ldots, x_n\}$. In $\mu x t$ all free occurrences of x in t are **bound** by μx. By $t[x/s]$ we denote the result of substituting all free occurrences of x in t by s, renaming bound variables of t to avoid capture of free variables of s by bindings in t. The **μ-depth** of a term is 0 for the terms x, 0, 1, is 1 plus the μ-depth of t for the term $\mu x t$, and is the maximum of the μ-depth of its immediate subterms, otherwise.

▶ **Definition 3.** A **partially ordered μ-semiring** $(M, +, \cdot, 0, 1, \leq)$ is a semiring $(M, +, \cdot, 0, 1)$ with a partial order $\leq$ on M, where every term t defines a function $t^M : (X \to M) \to M$, so that for all variables $x \in X$ and terms s, t we have:

1. for all valuations $g : X \to M$,

$$\begin{array}{llll} 0^M(g) = 0, & & (s+t)^M(g) = s^M(g) + t^M(g), \\ 1^M(g) = 1, & & (s \cdot t)^M(g) = s^M(g) \cdot g^M(g), \\ x^M(g) = g(x), & \text{if } s^M \leq t^M, & \text{then} & \mu x s^M \leq \mu x t^M, \end{array}$$

2. t^M is monotone with respect to the pointwise order on $X \to M$,
3. $t^M(g) = t^M(h)$, for all valuations $g, h : X \to M$ which agree on $\mathit{free}(t)$,
4. $t[x/s]^M(g) = t^M(g[x/s^M(g)])$, for all valuations $g : X \to M$.

When $\mathit{free}(t) \subseteq \{x_1, \ldots, x_n\}$ and $g(x_i) = a_i$ for $1 \leq i \leq n$, instead of $t^M(g)$ we often write $t^M[x_1/a_1, \ldots, x_n/a_n]$ or just $t^M(a_1, \ldots, a_n)$.

The final two conditions above are called the **coincidence** and **substitution properties**; in the latter, $g[x/a]$ denotes the valuation that agrees with g, except that it assigns a to x. Clearly, the substitution property extends to simultaneous substitutions $[x_1/s_1, \ldots, x_n/s_n]$. A first-order formula built from equations and inequations between μ-terms **holds in** M if it is true for every valuation $g : X \to M$.

▶ **Definition 4.** A **Park μ-semiring** is a partially ordered μ-semiring M where for all terms t and variables x, y, the following hold in M:

$$t[x/\mu x t] \quad \leq \quad \mu x t, \tag{1}$$

$$t[x/y] \leq y \quad \to \quad \mu x t \leq y. \tag{2}$$

It follows easily that $t[x/\mu x t] = \mu x t$ holds in M, as well as $\mu y.t[x/y] = \mu x t$, for $y \notin \mathit{free}(t)$.

Conditions (1) and (2) imply that $\mu x t^M(g)$ is the least solution of $t \leq x$ in M, g, i.e. the least $a \in M$ such that $t^M(g[x/a]) \leq a$.

3 Chomsky Algebras

▶ **Definition 5** ([11])**.** A **Chomsky algebra** $(M, +, \cdot, 0, 1)$ is an idempotent semiring where every finite system of polynomial inequations

$$\begin{array}{rcll} x_1 & \geq & p_1(x_1, \ldots, x_n, y_1, \ldots, y_m), & \\ & \vdots & & \\ x_n & \geq & p_n(x_1, \ldots, x_n, y_1, \ldots, y_m), & \text{abbreviated } \bar{x} \geq \bar{p}(\bar{x}, \bar{y}), \end{array} \tag{3}$$

has least solutions, i.e. for all $\bar{b} \in M^m$ there is a least $\bar{a} = a_1, \ldots, a_n \in M^n$ such that $a_i \geq p_i^M(\bar{a}, \bar{b})$ for $i = 1, \ldots, n$, where $\leq$ is the natural partial order on M defined by $a \leq b$ iff $a + b = b$. Of course, for each $\bar{b}$ the least solution $\bar{a}$ is unique.

▶ **Example 6.** Let $(X^*, \cdot, \epsilon)$ be the monoid of all finite words of elements of X, with concatenation $\cdot$ as product and the empty string ϵ as unit. Its power set $\mathcal{P}X^*$, the set of all languages over X, is an idempotent semiring $(\mathcal{P}X^*, +, \cdot, 0, 1)$, where $0 := \emptyset$, $1 := \{\epsilon\}$, and for $A, B \subseteq X^*$, $A + B := A \cup B$ is set union and $A \cdot B := \{a \cdot b \mid a \in A, b \in B\}$ the elementwise concatenation. A finite system of polynomial inequations (3) is a **context-free grammar** with nonterminals $x_1, \ldots, x_n$ and terminals $y_1, \ldots, y_m$. For a vector $\bar{B}$ of m languages, it leads to an increasing sequence $\bar{A}_k = (A_{k,1}, \ldots, A_{k,n})$ of language vectors by

$$A_{0,i} := \emptyset, \quad A_{k+1,i} := p_i^{\mathcal{P}X^*}(\bar{A}_k, \bar{B}), \quad i = 1, \ldots, n.$$

Clearly, any solution $\bar{A}$ of $\bar{x} \geq \bar{p}^{\mathcal{P}X^*}[\bar{y}/\bar{B}]$ must satisfy $\bar{A} \supseteq \bigcup\{\bar{A}_k \mid k \in \mathbb{N}\}$, where union and subsumption are meant componentwise. The least solution of the inequation system, relative to $\bar{B}$, is $\bar{A} := \bigcup\{\bar{A}_k \mid k \in \mathbb{N}\}$, since $+$ and $\cdot$ are compatible with arbitrary unions, i.e. for languages $A, B \subseteq X^*$ and $\emptyset \neq \mathcal{C} \subseteq \mathcal{P}X^*$,

$$\begin{array}{rcl} A + \bigcup \mathcal{C} & = & \bigcup\{A + C \mid C \in \mathcal{C}\}, \\ A \cdot \bigcup \mathcal{C} \cdot B & = & \bigcup\{A \cdot C \cdot B \mid C \in \mathcal{C}\}, \end{array}$$

which implies $\bar{A} \supseteq \bar{p}^{\mathcal{P}X^*}(\bar{A}, \bar{B})$. Therefore, $(\mathcal{P}X^*, \cup, \cdot, \emptyset, \{\epsilon\})$ is a Chomsky algebra.

Another example is the set $Rel\,(S)$ of all binary relations over a set S with empty relation as 0, the identity relation as 1, union as + and relation product as $\cdot$. The standard example of a Chomsky algebra is the algebra of context-free languages over the alphabet X.

▶ **Example 7.** The set $\mathcal{C}X^*$ of **context-free languages** over X is the smallest set $\mathcal{L} \subseteq \mathcal{P}X^*$ such that (i) each finite subset of $X \cup \{\epsilon\}$ is in $\mathcal{L}$ and (ii) if $\bar{x} \geq \bar{p}(\bar{x}, \bar{y})$ is a polynomial system, and $\bar{B} = B_1, \ldots, B_m \in \mathcal{L}$, then the components A_i of the least $\bar{A} = A_1, \ldots, A_n \in \mathcal{P}X^*$ with $\bar{A} \supseteq \bar{p}^{\mathcal{P}X^*}(\bar{A}, \bar{B})$ belong to $\mathcal{L}$. With the operations inherited from $\mathcal{P}X^*$, $(\mathcal{C}X^*, +, \cdot, 0, 1)$ is a Chomsky algebra. For example, $\{a^n b^n \mid n \in \mathbb{N}\}$ is a context-free language over $X \supseteq \{a, b\}$; it is the least solution of $x \geq axb + 1$ relative to the standard valuation $g(a) = \{a\}, g(b) = \{b\}$.

Of course, the regular languages over X do not form a Chomsky algebra, as they don't have solutions for inequations like $axb + 1 \leq x$.

The next lemma is a slight improvement of Lemma 2.1 in [11] in that it cares about the well-definedness of $\mu x t^M(g)$ and the partial order.

▶ **Lemma 8.** *Every Chomsky algebra M is an idempotent, partially ordered μ-semiring, if for all terms t, variables x and valuations $g : X \to M$ we take*

$$\mu x t^M(g) := \text{the least } a \in M \text{ such that } t^M(g[x/a]) \leq a. \tag{4}$$

Moreover, every inequation system $\bar{t}(\bar{x}, \bar{y}) \leq \bar{x}$ with μ-terms $\bar{t}(\bar{x}, \bar{y})$ has least solutions in M, i.e. for all parameters $\bar{b}$ from M there is a least tuple $\bar{a}$ in M such that $\bar{t}^M(\bar{a}, \bar{b}) \leq \bar{a}$.

Proof. We define term functions $t^M : (X \to M) \to M$ by induction on the μ-depth of t. The cases for terms of the forms 0, 1, x, $(s + t)$, and $(s \cdot t)$ are obvious; for $\mu x t$ we need to show that $t \leq x$ has least solutions in M, so that the definition of $\mu x t^M(g)$ via (4) is well-defined. More generally, we simultaneously prove by induction on k:

1. every inequation system $\bar{t} \leq \bar{x}$ has least solutions in M, for terms $\bar{t}$ of μ-depth $\leq k$,
2. term functions t^M for terms t of μ-depth $\leq k$ satisfy the properties of term functions in partially ordered μ-semirings (cf. Definition 3).

Let $\bar{t}(\bar{x}, \bar{y}) \leq \bar{x}$ be a system of term inequations

$$\begin{array}{rcl} t_1(x_1, \ldots, x_n, y_1, \ldots, y_m) & \leq & x_1 \\ & \vdots & \\ t_n(x_1, \ldots, x_n, y_1, \ldots, y_m) & \leq & x_n \end{array} \tag{5}$$

where each t_i has μ-depth $\leq k$. We may assume that bound variables in $\bar{t}(\bar{x}, \bar{y}) \leq \bar{x}$ are pairwise distinct and distinct from $\bar{x}, \bar{y}$. If $k = 0$, $\bar{t} \leq \bar{x}$ is a system of polynomial inequations and therefore has least solutions in M. If $k > 0$, we may assume that $t_n \leq x_n$ is an inequation where t_n is of maximal μ-depth. Then t_n can be written as $t_n = t'_n[x_{n+1}/\mu x_{n+1} t_{n+1}]$ where $\mu x_{n+1} t_{n+1}$ is of maximal μ-depth. Consider the inequation system $\bar{t}' \leq \bar{x}'$ obtained from $\bar{t} \leq \bar{x}$ by replacing $t_n \leq x_n$ by the two inequations $t'_n \leq x_n$ and $t_{n+1} \leq x_{n+1}$. Its maximal μ-depth, or its number of terms of maximal μ-depth, is less than that of $\bar{t} \leq \bar{x}$. Hence, by induction, $\bar{t}' \leq \bar{x}'$ has least solutions in M. Fix parameters $\bar{b}$ and let $(\bar{a}, a_{n+1})$ be the least elements of M such that $\bar{t}'^M(\bar{a}, a_{n+1}, \bar{b}) \leq (\bar{a}, a_{n+1})$. Then, by definition, $a_{n+1} = \mu x_{n+1} t_{n+1}{}^M(\bar{a}, \bar{b})$, hence by the substitution property for terms of μ-depth $\leq k$,

$$t_n^M(\bar{a}, \bar{b}) = t'_n[x_{n+1}/\mu x_{n+1} t_{n+1}]^M(\bar{a}, \bar{b}) = t'^M_n(\bar{a}, a_{n+1}, \bar{b}) \leq a_n.$$

This gives $\bar{t}^M(\bar{a}, \bar{b}) \leq \bar{a}$, and so $\bar{a}$ is a solution of $\bar{t} \leq x$ relative to $\bar{b}$. If $\bar{c}$ is another one, take $c_{n+1} := \mu x_{n+1} t_{n+1}{}^M(\bar{c}, \bar{b})$. Then $t^M_{n+1}(\bar{c}, c_{n+1}, \bar{b}) \leq c_{n+1}$ and, by the substitution

property, $t'^M_n(\bar{c}, c_{n+1}, \bar{b}) = t^M_n(\bar{c}, \bar{b}) \leq c_n$. Therefore, $\bar{t}'^M(\bar{c}, c_{n+1}, \bar{b}) \leq (\bar{c}, c_{n+1})$, whence $(\bar{a}, a_{n+1}) \leq (\bar{c}, c_{n+1})$ and $\bar{a} \leq \bar{c}$. Thus, $\bar{a}$ is the least solution of $\bar{t} \leq \bar{x}$ in M relative to $\bar{b}$.

We leave it to the reader to check that the properties of term functions in partially ordered μ-semirings hold in M for terms of μ-depth $\leq k$. ◀

▶ **Corollary 9.** *Under the interpretation of μ-terms in (4), every Chomsky algebra M is a Park μ-semiring. In particular, the context-free languages $\mathcal{C}X^*$ form a Park μ-semiring.*

Proof. Since $\mu x t^M(g)$ is the least $a \in M$ with $t^M(g[x/a]) \leq a$ and the substitution property holds in M, we have

$$t[x/\mu x t]^M(g) = t^M(g[x/\mu x t^M(g)]) \leq \mu x t^M(g).$$

Suppose $t[x/y]^M(g) \leq y^M(g) = g(y)$ for some $g : X \to M$ and some variable y. Then by the substitution property,

$$t^M(g[x/g(y)]) = t[x/y]^M(g) \leq g(y) = x^M(g[x/g(y)]),$$

so $g(y)$ is a solution of $t \leq x$ in M relative to g, and by comparison with the least solution, $\mu x t^M(g) \leq g(y)$. ◀

▶ **Lemma 10.** *If $g : X \to \mathcal{C}X^*$, then $t^{\mathcal{C}X^*}(g) = t^{\mathcal{P}X^*}(g)$ for each μ-term t.*

Proof. By induction on the structure of t. For $\mu x t$, let $\bar{x} \geq \bar{p}_t(\bar{x}, \bar{y})$ be the polynomial system obtained from $t(x, \bar{y})$ in the proof of Lemma 8 such that

$$\mu x t^{\mathcal{P}X^*}(g) = \text{the least } A \subseteq X^* \text{ such that } t^{\mathcal{P}X^*}(g[x/A]) \subseteq A \tag{6}$$

is a component of the least $\bar{A}$ in $\mathcal{P}X^*$ with $\bar{A} \supseteq \bar{p}_t^{\mathcal{P}X^*}(\bar{A}, \bar{B})$ and $\bar{B} = g(\bar{y})$. By definition of $\mathcal{C}X^*$, $\bar{A}$ and hence $A := \mu x t^{\mathcal{P}X^*}(g)$ belong to $\mathcal{C}X^*$. By induction,

$$t^{\mathcal{C}X^*}(g[x/A]) = t^{\mathcal{P}X^*}(g[x/A]) \subseteq A,$$

so by Park's axioms, $\mu x t^{\mathcal{C}X^*}(g) \subseteq A = \mu x t^{\mathcal{P}X^*}(g)$. The converse $\mu x t^{\mathcal{P}X^*}(g) \subseteq \mu x t^{\mathcal{C}X^*}(g)$ follows from the induction hypothesis and the fact that $\mathcal{C}X^* \subseteq \mathcal{P}X^*$. ◀

4 Vector Versions

To denote the least simultaneous fixed point of an inequation system $\bar{x} \geq \bar{t}$, one might introduce terms $\mu \bar{x} \bar{t}$ where μ is an n-ary fixed-point operator, accompanied by projection functions π^n_i to get the i-th component of the fixed point. As has been observed by Bekić[1], de Bakker, Scott around 1970, it is sufficient to have a unary fixed point operator. Components of a simultaneous fixed point can be named by terms using the unary μ in a nested fashion.

▶ **Theorem 11** (Bekić [1])**.** *Let $(M, \leq)$ be a partially ordered set in which every countable increasing chain $\{a_i \mid i \in \mathbb{N}\}$ has a least upper bound, $\sum_{i \in \mathbb{N}} a_i$. Suppose $f, g : M^2 \to M$ are* **continuous** *in each component, i.e. map the least upper bound of countable increasing chains to the least upper bound of the images of the chain elements. Then the least solution of the system $(x, y) \geq (f(x, y), g(x, y))$ can be obtained by taking the least solution of each inequation separately, plugging it into the other equation and taking the least solutions of the resulting equations separately. I.e., the binary and unary fixed point operators are related by*

$$\mu(x, y)(f(x, y), g(x, y)) = (\mu x.f(x, \mu y.g(x, y)), \mu y.g(\mu x.f(x, y), y). \tag{7}$$

For an n-dimensional inequation system $\bar{x} \geq \bar{t}$, we define an n-tuple $\mu\bar{x}\bar{t}$ of μ-terms by recursively using Bekić's equations (7).

▶ **Definition 12.** ([10]) For vectors $\bar{t} = t_1, \ldots, t_n$ of terms and $\bar{x} = x_1, \ldots, x_n$ of pairwise different variables, we define the term vector $\mu\bar{x}\bar{t}$ as follows. If $n = 1$, then $\mu\bar{x}\bar{t} := \mu x_1 t_1$. If $n > 1$, $\bar{x} = (\bar{y}, \bar{z})$ and $\bar{t} = (\bar{r}, \bar{s})$ with term vectors $\bar{r}, \bar{s}$ of lengths $|\bar{y}|, |\bar{z}| < n$, then $\mu\bar{x}\bar{t}$ is[1]

$$\mu(\bar{y}, \bar{z})(\bar{r}, \bar{s}) := (\mu\bar{y}.\bar{r}[\bar{z}/\mu\bar{z}\bar{s}], \mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]). \tag{8}$$

A Chomsky algebra need not be closed under unions of countable increasing chains. For example, the set $\mathcal{C}X^*$ of all context-free languages over X has increasing chains of finite languages whose unions are not context-free. So one cannot apply Bekić's theorem literally to prove that $\mu\bar{x}\bar{t}$ denotes the least solution of $\bar{x} \geq \bar{t}$ in Chomsky algebras. Instead, we prove the Park axioms for term vectors by induction on the dimension.

For term vectors $\bar{s}$, $\bar{t}$ of the same dimension, let $\bar{s} = \bar{t}$ resp. $\bar{s} \leq \bar{t}$ be the conjunction of the equations resp. inequations of corresponding components. For $\bar{t} = (t_1, \ldots, t_n)$ we write $\bar{t}^M(g)$ for $(t_1^M(g), \ldots, t_n^M(g))$. The following property can be shown by induction on $|\bar{x}|$.

▶ **Lemma 13.** *If none of the variables in $\bar{x}$ occurs free in the terms of $\bar{s}$, then*

$$\mu\bar{x}\bar{t}[\bar{y}/\bar{s}] = \mu\bar{x}.\bar{t}[\bar{y}/\bar{s}],$$

A proof of this and the next lemma is given in the appendix.

▶ **Lemma 14.** *Let M be a Park μ-semiring. For all vectors $\bar{t}$ of terms and vectors $\bar{x}, \bar{y}$ of variables of the same dimension, the vector versions of (1) and (2),*

$$\bar{t}[\bar{x}/\mu\bar{x}\bar{t}] \quad \leq \quad \mu\bar{x}\bar{t}, \tag{9}$$

$$\bar{t}[\bar{x}/\bar{y}] \leq \bar{y} \quad \to \quad \mu\bar{x}\bar{t} \leq \bar{y}, \tag{10}$$

hold in M. Hence, for any Chomsky algebra M and valuation $g : X \to M$, $\mu\bar{x}\bar{t}^M(g)$ is the least $\bar{a}$ such that $\bar{t}^M(g[\bar{x}/\bar{a}]) \leq \bar{a}$.

One shows (9) and (10) simultaneously with a vector version of the substitution property, using lemma 13: in any Park μ-seminiring M and for any $g : X \to M$,

$$\mu\bar{x}\bar{t}[\bar{y}/\bar{s}]^M(g) = \mu\bar{x}\bar{t}^M(g[\bar{y}/\bar{s}^M(g)]), \qquad \text{if no variable of } \bar{x} \text{ is free in the terms } \bar{s}. \tag{11}$$

▶ **Corollary 15.** *If M is a Park μ-semiring, the vector version of the μ-rule holds: for vectors $\bar{s}$, $\bar{t}$ of terms and $\bar{x}$ of different variables, all of the same dimension, if $\bar{s}^M \leq \bar{t}^M$, then $\mu\bar{x}\bar{s}^M \leq \mu\bar{x}\bar{t}^M$.*

5 μ-Continuity

The usual way to compute the simultaneous least fixed point $\bar{A}$ of $\bar{x} \geq \bar{t}(\bar{x}, \bar{y})$ in $\mathcal{P}X^*$ relative to $\bar{B}$ is to approximate it by its finite stages $\bar{A}_m$ and (componentwise) take their union, i.e.

$$\bar{A} := \bigcup_{m \in \mathbb{N}} \bar{A}_m, \qquad \text{where} \quad \bar{A}_0 := \bar{\emptyset}, \qquad \bar{A}_{m+1} := \bar{t}^{\mathcal{P}X^*}(\bar{A}_m, \bar{B}).$$

As shown below, the continuity of $+$ and $\cdot$ in $\mathcal{P}X^*$ imply that $\bar{A}$ equals $\mu\bar{x}\bar{t}^{\mathcal{P}X^*}(\bar{B})$.

[1] To distinguish $\mu xt[y/s]$ from $\mu x(t[y/s])$, we write $\mu x.t[y/s]$ for the latter, using . to save the brackets of the metalanguage. We have no . in the object language's μ-terms, preferring $\mu x(t + s)$ over $\mu x.t + s$.

▶ **Lemma 16.** *In $M = \mathcal{P}X^*$ or $M = \mathcal{C}X^*$, all term functions are* **continuous***: for any term t, valuation $g : X \to M$ and increasing chain $\{A_i \mid i \in \mathbb{N}\}$ of elements with union in M,*

$$t^M(g[y/\bigcup_{i\in\mathbb{N}} A_i]) = \bigcup_{i\in\mathbb{N}} t^M(g[y/A_i]). \tag{12}$$

Proof. This is clear when t is one of $x, y, 0, 1$. For $(t_1 + t_2)$ and $(t_1 \cdot t_2)$ use induction, the continuity of $+$ and $\cdot$ in M, and the fact that the A_i form an increasing chain. For μxt, by the monotonicity of μxt^M it is sufficient to show

$$\mu xt^M(g[y/\bigcup_{i\in\mathbb{N}} A_i]) \subseteq \bigcup_{i\in\mathbb{N}} \mu xt^M(g[y/A_i]). \tag{13}$$

Let $B_i := \mu xt^M(g[y/A_i])$ for $i \in \mathbb{N}$, and let A and B be the unions of the increasing chains $\{A_i \mid i \in \mathbb{N}\}$ and $\{B_i \mid i \in \mathbb{N}\}$, respectively. If $M = \mathcal{P}X^*$, then $B \in M$ and

$$\begin{array}{rcll} t^M(g[y/A][x/B]) & = & \bigcup_{i,j\in\mathbb{N}} t^M(g[y/A_i][x/B_j]) & \text{(by induction)} \\ & = & \bigcup_{i\in\mathbb{N}} t^M(g[y/A_i][x/B_i]) & \text{(by monotonicity, increasing chains)} \\ & = & \bigcup_{i\in\mathbb{N}} t[x/\mu xt]^M(g[y/A_i]) & \text{(by the substitution property)} \\ & \subseteq & \bigcup_{i\in\mathbb{N}} \mu xt^M(g[y/A_i]) & \text{(by Park's inequation)} \\ & = & \bigcup_{i\in\mathbb{N}} B_i = B. & \end{array}$$

By the Park rule, $\mu xt^M(g[y/A]) \subseteq B$ follows, which proves (13) for $M = \mathcal{P}X^*$. By lemma 10, we can transfer (12) from $M = \mathcal{P}X^*$ to $M = \mathcal{C}X^*$. ◀

The power set semiring $(\mathcal{P}M, \cup, \cdot, \emptyset, \{1\})$ of a monoid $(M, \cdot^M, 1)$, a generalization of Example 6, and the semiring $Rel\,(S)$ of all binary relations on a set S are **continuous**: their operations $+$ and $\cdot$ are continuous and the partial order $\subseteq$ is complete (all directed subsets have suprema). The semiring $\mathcal{C}X^*$ of context-free languages over X is *not* complete.

▶ **Definition 17** ([11])**.** A Chomsky algebra M is **μ-continuous**, if for all μ-terms $t(x, \bar{y})$ and all $a, b, \bar{c} \in M$,

$$a \cdot \mu xt^M[\bar{y}/\bar{c}] \cdot b = \sum\{a \cdot (mxt)^M[\bar{y}/\bar{c}] \cdot b \mid m \in \mathbb{N}\}, \tag{14}$$

where the term mxt, the m-fold iteration of t in x, is defined by $0xt := 0$, $(m+1)xt := t[x/mxt]$. In particular, the supremum on the right hand side of (14) must exist.

As has been mentioned in [11] without proof, we have:

▶ **Theorem 18.** *The Chomsky algebra $\mathcal{C}X^*$ of all context-free languages over X is μ-continuous. In particular, for all terms t and valuations $g : X \to \mathcal{C}X^*$,*

$$\mu xt^{\mathcal{C}X^*}(g) = \bigcup\{mxt^{\mathcal{C}X^*}(g) \mid m \in \mathbb{N}\}.$$

Proof. We first consider $M = \mathcal{P}X^*$ and then transfer the result to $\mathcal{C}X^*$ by Lemma 10. From $0xt^M(g) \subseteq 1xt^M(g)$ and $0xt^M(g) \subseteq \mu xt^M(g)$, by induction we get $mxt^M(g) \subseteq (m+1)xt^M(g) \subseteq \mu xt^M(g)$ for all m, using monotonicity, the substitution property, and the Park inequation. Therefore, $\bigcup\{mxt^M(g) \mid m \in \mathbb{N}\} \subseteq \mu xt^M(g)$. For the converse,

$$\begin{array}{rcll} t^M(g[x/\bigcup\{(mxt)^M(g) \mid m \in \mathbb{N}\}]) & = & \bigcup\{t^M(g[x/(mxt)^M(g)]) \mid m \in \mathbb{N}\} & \text{(by (12))} \\ & = & \bigcup\{t[x/mxt]^M(g) \mid m \in \mathbb{N}\} & \\ & = & \bigcup\{((m+1)xt)^M(g) \mid m \in \mathbb{N}\} & \\ & = & \bigcup\{(mxt)^M(g) \mid m \in \mathbb{N}\}. & \end{array}$$

Hence, by the Park rule, $\mu xt^M(g) \subseteq \bigcup\{(mxt)^M(g) \mid m \in \mathbb{N}\}$. It follows that for $A, B \in M$, $A \cdot \mu xt^M(g) \cdot B = \bigcup\{A \cdot mxt^M(g) \cdot B \mid m \in \mathbb{N}\}$. ◀

The iterative computation of least fixed points is used in [11] to interpret μ-terms in the semiring $\mathcal{C}X^*$:

▶ **Example 19.** ([11]) The **canonical interpretation** L of μ-terms over X in $\mathcal{C}X^*$ is

$$\begin{array}{lcllcl} L(x) & = & \{x\} & L(s+t) & = & L(s) \cup L(t) \\ L(0) & = & \emptyset & L(s \cdot t) & = & \{uv \mid u \in L(s), v \in L(t)\} \\ L(1) & = & \{\epsilon\} & L(\mu x t) & = & \bigcup\{L(mxt) \mid m \in \mathbb{N}\}. \end{array}$$

By Theorem 18, the canonical interpretation L of μ-terms in the semiring $\mathcal{C}X^*$ coincides with the interpretation in the Chomsky algebra $\mathcal{C}X^*$ under the valuation L, i.e. $L(t) = t^{\mathcal{C}X^*}(L)$ for all terms t. This can be proven by induction on the well-founded relation $\prec$ of terms from [7] were $t_i \prec (t_1 + t_2)$, $t_i \prec (t_1 \cdot t_2)$, and $mxt \prec \mu x t$.

▶ Remark. Grathwohl e.a. [11], Theorem 3.2, prove that an idempotent semiring M with an interpretation of μ-terms satisfying the μ-continuity condition also satisfies the Park axioms, hence makes $\mu x t^M(g)$ the least fixed point of $x \geq t$ in M relative to g.

For a vector of terms $\bar{t} = (t_1, \ldots, t_n)$, put $L(\bar{t}) := (L(t_1), \ldots, L(t_n))$. The m-th iteration of $\bar{t}$ in $\bar{x}$ can be expressed syntactically by a term vector $m\bar{x}\bar{t}$, where $|\bar{x}| = |\bar{t}|$ and

$$0\bar{x}\bar{t} := \bar{0}, \qquad (m+1)\bar{x}\bar{t} := \bar{t}[\bar{x}/m\bar{x}\bar{t}]. \tag{15}$$

Vector versions of substitution and other properties of term functions in $\mathcal{C}X^*$ will be used for L, based on (11) and theorem 18.

We now prove a vector version of (the main part of) the μ-continuity condition for $\mathcal{C}X^*$, i.e. that the least fixed point of a system $\bar{x} \geq \bar{t}$, as determined by the Bekić-Scott equations embodied in $\mu\bar{x}\bar{t}$, is the supremum of its approximations by iterations $m\bar{x}\bar{t}$. In particular, Bekić's reduction works in $\mathcal{C}X^*$, where not every ascending chain has a supremum.

▶ **Lemma 20.** *For all vectors $\bar{x}, \bar{y}, \bar{z}$ of pairwise distinct variables and vectors $\bar{t}, \bar{r}, \bar{s}$ of μ-terms such that $|\bar{x}| = |\bar{t}|$, $|\bar{y}| = |\bar{r}|$ and $|\bar{z}| = |\bar{s}|$, we have:*
1. $L(\mu\bar{x}\bar{t}) = \bigcup\{L(m\bar{x}\bar{t}) \mid m \in \mathbb{N}\}$,
2. $L(\bar{s}[\bar{x}/\mu\bar{x}\bar{t}]) = \bigcup\{L(\bar{s}[\bar{x}/m\bar{x}\bar{t}]) \mid m \in \mathbb{N}\}$,
3. $L(\mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]) = \bigcup\{L(m\bar{z}.\bar{s}[\bar{y}/k\bar{y}\bar{r}]) \mid m, k \in \mathbb{N}\}$.

Proof. Proof by simultaneous induction on the vector length of $|\bar{x}| = |\bar{y}| + |\bar{z}|$ with $|\bar{y}|, |\bar{z}| < |\bar{x}|$.
1. For $|\bar{x}| = 1$, the claim holds by definition of L, so suppose $|\bar{x}| > 1$.

(a) $\bigcup_m L(m\bar{x}\bar{t}) \subseteq L(\mu\bar{x}\bar{t})$: Clearly, $L(0\bar{x}\bar{t}) = \bar{\emptyset} \subseteq L(\mu\bar{x}\bar{t})$. If $L(m\bar{x}\bar{t}) \subseteq L(\mu\bar{x}\bar{t})$, then

$$\begin{array}{lcll} L((m+1)\bar{x}\bar{t}) & = & L(\bar{t}[\bar{x}/m\bar{x}\bar{t}]) & ((15)) \\ & = & L[\bar{x}/L(m\bar{x}\bar{t})](\bar{t}) & \text{(substitution property)} \\ & \subseteq & L[\bar{x}/L(\mu\bar{x}\bar{t})](\bar{t}) & \text{(induction hypothesis)} \\ & = & L(\bar{t}[\bar{x}/\mu\bar{x}\bar{t}]) & \text{(substitution property)} \\ & \subseteq & L(\mu\bar{x}\bar{t}) & \text{(Lemma 14, (9)).} \end{array}$$

(b) $L(\mu\bar{x}\bar{t}) \subseteq \bigcup\{L(m\bar{x}\bar{t}) \mid m \in \mathbb{N}\}$. By induction, claim 3. holds with $|\bar{y}|, |\bar{z}| < |\bar{x}|$, so

$$\begin{array}{lcll} L(\mu\bar{x}\bar{t}) & = & L(\mu(\bar{y}, \bar{z})(\bar{r}, \bar{s})) & (\bar{x} = (\bar{y}, \bar{z}), \bar{t} = (\bar{r}, \bar{s})) \\ & = & L(\mu\bar{y}.\bar{r}[\bar{z}/\mu\bar{z}\bar{s}], \mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]) & \text{(definition of } \mu\bar{x}\bar{t}) \\ & = & (\bigcup_{m,k} L(m\bar{y}.\bar{r}[\bar{z}/k\bar{z}\bar{s}]), \bigcup_{m,k} L(m\bar{z}.\bar{s}[\bar{y}/k\bar{y}\bar{r}])) & \text{(induction hypothesis 3.)} \\ & = & \bigcup_{m,k} L(m\bar{y}.\bar{r}[\bar{z}/k\bar{z}\bar{s}], m\bar{z}.\bar{s}[\bar{y}/k\bar{y}\bar{r}]) & \text{(monotonicity).} \end{array}$$

It is therefore sufficient to show

$$L(m\bar{y}.\bar{r}[\bar{z}/k\bar{z}\bar{s}], m\bar{z}.\bar{s}[\bar{y}/k\bar{y}\bar{r}]) \subseteq L(m_k(\bar{y},\bar{z})(\bar{r},\bar{s})), \quad \text{for } m_k = m(k+1). \tag{16}$$

Put $(\bar{A}_n, \bar{B}_n) := L(n\bar{x}\bar{t}) = L(n(\bar{y},\bar{z})(\bar{r},\bar{s}))$. By induction on k one obtains

$$L[\bar{y}/\bar{A}_n](k\bar{z}\bar{s}) \subseteq \bar{B}_{n+k} \quad \text{and} \quad L[\bar{z}/\bar{B}_n](k\bar{y}\bar{r}) \subseteq \bar{A}_{n+k}, \tag{17}$$

using monotonicity of $\bar{s}^{\mathcal{C}X^*}$ and $\bar{r}^{\mathcal{C}X^*}$ and the substitution property. To see (16), put $\bar{r}_k(\bar{y}) := \bar{r}[\bar{z}/k\bar{z}\bar{s}]$ and $\bar{s}_k(\bar{z}) := \bar{s}[\bar{y}/k\bar{y}\bar{r}]$. Suppose that for some m we have

$$L(m\bar{y}\bar{r}_k, m\bar{z}\bar{s}_k) \subseteq (\bar{A}_n, \bar{B}_n) \qquad \text{with } n = m(k+1). \tag{18}$$

which is clearly true for $m = 0$. Then, using the monotonicity of $\bar{r}^{\mathcal{C}X^*}$ and $\bar{s}^{\mathcal{C}X^*}$,

$$\begin{array}{lll}
L((m+1)\bar{y}\bar{r}_k, (m+1)\bar{z}\bar{s}_k) & & \\
\quad = & L(\bar{r}_k[\bar{y}/m\bar{y}\bar{r}_k], \bar{s}_k[\bar{z}/m\bar{z}\bar{s}_k]) & \text{(definition)} \\
\quad \subseteq & (L[\bar{y}/\bar{A}_n](\bar{r}_k), L[\bar{z}/\bar{B}_n](\bar{s}_k)) & \text{(substitution, (18))} \\
\quad \subseteq & (L[\bar{y}/\bar{A}_n, \bar{z}/\bar{B}_{n+k}](\bar{r}), L[\bar{y}/\bar{A}_{n+k}, \bar{z}/\bar{B}_n](\bar{s})) & \text{((17))} \\
\quad \subseteq & (L[\bar{y}/\bar{A}_{n+k}, \bar{z}/\bar{B}_{n+k}](\bar{r}), L[\bar{y}/\bar{A}_{n+k}, \bar{z}/\bar{B}_{n+k}](\bar{s})) & \text{(increasing chains)} \\
\quad = & (\bar{A}_{n+k+1}, \bar{B}_{n+k+1}) & \\
\quad = & (\bar{A}_{(m+1)(k+1)}, \bar{B}_{(m+1)(k+1)}). &
\end{array}$$

Thus, we have (16), and hence $L(\mu\bar{x}\bar{t}) \subseteq \bigcup\{L(m\bar{x}\bar{t}) \mid m \in \mathbb{N}\}$.

2. This is a vector version of lemma 2.5 of [11]. Since $\bigcup$ on $(\mathcal{C}X^*)^{|\bar{t}|}$, substitution $\bar{s}[\bar{x}/\mu\bar{x}\bar{t}]$ and evaluation $L(\bar{s})$ is done componentwise, it is sufficient to consider $|\bar{s}| = 1$.
By 1., $L(\mu\bar{x}\bar{t}) = \bigcup\{\bar{A}_m \mid m \in \mathbb{N}\}$, where $\bar{A}_m = L(m\bar{x}\bar{t})$ is a vector of (definable, hence) context-free languages. By induction, $\bar{A}_m \subseteq \bar{A}_{m+1}$ for all m, componentwise. Then

$$\begin{array}{lll}
L(s[\bar{x}/\mu\bar{x}\bar{t}]) & = & L[\bar{x}/L(\mu\bar{x}\bar{t})](s) \qquad \text{(substitution property)} \\
& = & L[\bar{x}/\bigcup\{L(m\bar{x}\bar{t}) \mid m \in \mathbb{N}\}](s) \qquad \text{(claim 1.)} \\
& = & \bigcup\{L[\bar{x}/L(m\bar{x}\bar{t})](s) \mid m \in \mathbb{N}\} \qquad \text{(Lemma 16)} \\
& = & \bigcup\{L(s[\bar{x}/m\bar{x}\bar{t}]) \mid m \in \mathbb{N}\} \qquad \text{(substitution property)}
\end{array}$$

3. To prove claim 1. for $|\bar{x}| > 1$, we used claim 3. with $|\bar{y}| + |\bar{z}| = |\bar{x}|$ and $0 < |\bar{y}|, |\bar{z}| < |\bar{x}|$. Hence, to prove claim 3. for dimension $|\bar{y}| + |\bar{z}|$, we have claim 1. for smaller dimensions as induction hypothesis. Therefore, an application of 1. (with $\mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]$ as $\mu\bar{x}\bar{t}$) gives

$$L(\mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]) = \bigcup_m L(m\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]).$$

To finish the proof of claim 3., it is now sufficient to show for all m:

$$L(m\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]) = \bigcup_k L(m\bar{z}.\bar{s}[\bar{y}/k\bar{y}\bar{r}]). \tag{19}$$

Equation (19) is clear for $m = 0$. Assume it is true for m. Then we proceed as follows, using abbreviations $\bar{s}_\omega := \bar{s}[\bar{y}/\mu\bar{y}\bar{r}]$ and $\bar{s}_k := \bar{s}[\bar{y}/k\bar{y}\bar{r}]$:

$$\begin{array}{lll}
L((m+1)\bar{z}\bar{s}_\omega) & = & L(\bar{s}_\omega[\bar{z}/m\bar{z}\bar{s}_\omega]) \qquad \text{(definition)} \\
& = & L[\bar{z}/L(m\bar{z}\bar{s}_\omega)](\bar{s}_\omega) \qquad \text{(substitution property)} \\
& = & \bigcup_k L[\bar{z}/L(m\bar{z}\bar{s}_\omega)](\bar{s}_k) \qquad \text{(induction hypothesis 2. for } |\bar{y}|) \\
& = & \bigcup_k L[\bar{z}/\bigcup_{k'} L(m\bar{z}\bar{s}_{k'})](\bar{s}_k) \qquad \text{(induction hypothesis (19))} \\
& = & \bigcup_k \bigcup_{k'} L[\bar{z}/L(m\bar{z}\bar{s}_{k'})](\bar{s}_k) \qquad \text{(Lemma 16, componentwise)} \\
& = & \bigcup_k L[\bar{z}/L(m\bar{z}\bar{s}_k)](\bar{s}_k) \qquad \text{(monotonicity)} \\
& = & \bigcup_k L(\bar{s}_k[\bar{z}/m\bar{z}\bar{s}_k]) \qquad \text{(substitution property)} \\
& = & \bigcup_k L((m+1)\bar{z}\bar{s}_k) \qquad \text{(definition).}
\end{array}$$

Hence we have (19) for all m. ◀

By the following lemma from [11] we can transfer the vector version of the μ-continuity condition from L resp. $\mathcal{C}X^*$ to an arbitrary μ-continuous Chomsky algebra.

▶ **Lemma 21** ([11], Lemma 3.1)**.** *Let M be a μ-continuous Chomsky algebra, $g : X \to M$ a valuation of terms in M, $h : X \to \mathcal{C}X^*$ be a valuation in the algebra $\mathcal{C}X^*$ of context-free languages over X, such that, for all $x \in X$ and μ-terms s, u,*

$$(sxu)^M(g) = \sum\{(syu)^M(g) \mid y \in h(x)\}.$$

Then, for all μ-terms s, t, u,

$$(stu)^M(g) = \sum\{(syu)^M(g) \mid y \in t^{\mathcal{C}X^*}(h)\}.$$

Notice that the canonical interpretation $L : X \to \mathcal{C}X^*$ satisfies the assumptions on the valuation h of the lemma.

▶ **Corollary 22.** *For any vectors $\bar{s}, \bar{t}, \bar{u}$ of μ-terms of equal length, any μ-continuous Chomsky-algebra M and valuation $g : X \to M$, under componentwise multiplication and supremum,*

$$(\bar{s} \cdot \bar{t} \cdot \bar{u})^M(g) = \sum\{(\bar{s} \cdot \bar{w} \cdot \bar{u})^M(g) \mid \bar{w} \in L(\bar{t})\}, \tag{20}$$

where, for $x_1 \cdots x_k \in X^*$, $(x_1 \cdots x_k)^M(g) := g(x_1) \cdot^M \ldots \cdot^M g(x_k)$.

This is clear since for each component, the equation follows from the lemma. The vector-version of the μ-continuity condition follows:

▶ **Corollary 23.** *Let M be a μ-continuous Chomsky algebra and $g : X \to M$. Then*

$$\bar{a} \cdot \mu\bar{x}\bar{t}^M(g) \cdot \bar{b} = \sum\{\bar{a} \cdot m\bar{x}\bar{t}^M(g) \cdot \bar{b} \mid m \in \mathbb{N}\},$$

for any term vector $\bar{t}$ and vectors $\bar{a}, \bar{b}$ of elements of M of the same length as $\bar{t}$.

Proof. We may assume that there are suitable vectors $\bar{s}, \bar{u}$ of terms such that $\bar{a} = \bar{s}^M(g)$ and $\bar{b} = \bar{u}^M(g)$. Then

$$\begin{array}{rcll}
\bar{a} \cdot \mu\bar{x}\bar{t}^M(g) \cdot \bar{b} & = & (\bar{s} \cdot \mu\bar{x}\bar{t} \cdot \bar{u})^M(g) & \\
& = & \sum\{(\bar{s} \cdot \bar{w} \cdot \bar{u})^M(g) \mid \bar{w} \in L(\mu\bar{x}\bar{t})\} & \text{(equation (20))} \\
& = & \sum\{(\bar{s} \cdot \bar{w} \cdot \bar{u})^M(g) \mid \bar{w} \in \bigcup\{L(m\bar{x}\bar{t}) \mid m \in \mathbb{N}\}\} & \text{(Lemma 20, 1.)} \\
& = & \sum\bigcup\{\{(\bar{s} \cdot \bar{w} \cdot \bar{u})^M(g) \mid \bar{w} \in L(m\bar{x}\bar{t})\} \mid m \in \mathbb{N}\} & \\
& = & \sum\{\sum\{(\bar{s} \cdot \bar{w} \cdot \bar{u})^M(g) \mid \bar{w} \in L(m\bar{x}\bar{t})\} \mid m \in \mathbb{N}\} & \\
& = & \sum\{(\bar{s} \cdot m\bar{x}\bar{t} \cdot \bar{u})^M(g) \mid m \in \mathbb{N}\} & \text{(equation (20))} \\
& = & \sum\{\bar{a} \cdot m\bar{x}\bar{t}^M(g) \cdot \bar{b} \mid m \in \mathbb{N}\}. &
\end{array}$$

◀

In particular, μ-continuity of M implies that the least solution $\mu\bar{x}\bar{t}^M(g)$ of a system $\bar{t} \leq \bar{x}$ relative to g is the supremum of the finite iterations $m\bar{x}\bar{t}^M(g)$:

▶ **Corollary 24.** *Let M be a μ-continuous Chomsky algebra and $g : X \to M$. Then*

$$\mu\bar{x}\bar{t}^M(g) = \sum\{m\bar{x}\bar{t}^M(g) \mid m \in \mathbb{N}\}.$$

6 Closure under Matrix Rings

To prove the main result, that the square matrices over a μ-continuous Chomsky algebra form a μ-continuous Chomsky algebra, we recall the case of Park μ-semirings:

▶ **Theorem 25** ([10], Theorem 7.6). *If M is a Park μ-semiring, so is $Mat_{n,n}(M)$.*

Proof. Let M be a Park μ-semiring and $N = Mat_{n,n}(M)$ the set of $n \times n$ matrices of elements of M. Equipped with the usual matrix operations, $(N, +, \cdot, 0, 1)$ is a semiring, because M is. To define the term functions $t^N : (X \to N) \to N$, for each variable $x \in X$ we fix n^2 distinct variables $x_{i,j}$, $1 \le i, j \le n$, which also have to be distinct from all $y_{i,j}$ for variables $y \neq x$. For each term t, define a vector t' of n^2 terms recursively by

$$\begin{array}{lcllcl} x' & := & (x_{i,j}), & (s+t)' & := & s' + t', \\ 0' & := & 0_{n,n}, & (s \cdot t)' & := & s' \cdot t', \\ 1' & := & 1_{n,n}, & (\mu x t)' & := & \mu x' t', \end{array}$$

where $0_{n,n}$ and $1_{n,n}$ are the zero and unit matrices, $+$ and $\cdot$ are the usual matrix addition and multiplication operations applied to matrices of terms, and $\mu x' t'$ is the term vector $\mu \bar{x} \bar{t}$ defined recursively by Bekić's equation (8) from x' and t'.

Any valuation $g : X \to N$ is obtained from a valuation $\hat{g} : X \to M$ by $g(x) = (a_{i,j})$, where $a_{i,j} = \hat{g}(x_{i,j})$ for $1 \le i, j \le n$. Define the term function t^N by

$$t^N(g) := ({t'_{i,j}}^M(\hat{g})), \qquad \text{where } t'_{i,j} \text{ is the } (i,j)\text{-th entry of } t'. \tag{21}$$

Concerning the properties for partially ordered μ-semirings, those for 0^N, 1^N, x^N, $(s+t)^N$ and $(s \cdot t)^N$ are immediate from the definition. The μ-rule holds in N, since, by corollary 15, the vector version of the μ-rule holds in M.

The monotonicity of the term function t^N follows from the monotonicity of the term functions ${t'}^M_{i,j}$. Likewise for the coincidence property. By Lemma 13 and Lemma 14, the vector versions of the substitution property and Park axioms hold in M; these imply that the substitution property and the Park axioms hold in N. ◀

▶ **Theorem 26.** *If M is a (μ-continuous) Chomsky algebra, so is $Mat_{n,n}(M)$.*

Proof. Let M be a Chomsky algebra and $N := Mat_{n,n}(M)$. By Corollary 9, M is an idempotent Park μ-semiring. Hence, by Theorem 25, N also is, and thus, by Lemma 14, N satisfies the vector versions of Park's axioms. In particular, every system $\bar{x} \ge \bar{p}(\bar{x}, \bar{y})$ of polynomial inequations does have, for any parameters $\bar{B} \in N$, a least solution in N, $(\mu \bar{x} \bar{p})^N(\bar{B})$. Hence, N is a Chomsky algebra.

Suppose, in addition, M is μ-continuous. To show that N is μ-continuous, we assume $n > 1$, as $Mat_{1,1}(M)$ is isomorphic to M. Let $A = (a_{i,j}), B = (b_{i,j}) \in N$, $t(x, \bar{y})$ a μ-term and $g : X \to N$ coming from a valuation $\hat{g} : X \to M$ as in (21). In order to show

$$A \cdot \mu x t^N(g) \cdot B = \sum \{A \cdot m x t^N(g) \cdot B \mid m \in \mathbb{N}\}, \tag{22}$$

we first show

$$\mu x t^N(g) = \sum \{m x t^N(g) \mid m \in \mathbb{N}\}. \tag{23}$$

By definition, $\mu x t^N(g) = ((\mu x t)'^M_{i,j}(\hat{g}))$, where $(\mu x t)' = \mu x' t'$ is obtained from the matrices

$$x' = \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,n} \end{pmatrix} \quad \text{and} \quad t' = \begin{pmatrix} t_{1,1} & \dots & t_{1,n} \\ \vdots & \ddots & \vdots \\ t_{n,1} & \dots & t_{n,n} \end{pmatrix}$$

of pairwise different variables $x_{i,j}$ and of μ-terms $t_{i,j}(x', \bar{y}')$ (with variables $y' = (y_{i,j})$ for parameters y in $\bar{y}$) according to Definition 12 for the inequation system $t' \leq x'$ of size n^2. Accordingly, we have a square term matrix $(mxt)' = mx't'$ for the m-th iteration of t' in x'. By Corollary 24, we get (23):

$$\mu xt^N(g) = ((\mu xt)'^M_{i,j}(\hat{g})) = \sum\{((mxt)'_{i,j}{}^M(\hat{g})) \mid m \in \mathbb{N}\} = \sum\{mxt^N(g) \mid m \in \mathbb{N}\}.$$

Concerning (22), it is sufficient to consider the (i,j)-th entry and use Corollary 23:

$$\begin{aligned}
&(A \cdot \mu xt^N(g) \cdot B)_{i,j} \\
&= \sum_{k,l \leq n} a_{i,k} \cdot^M (\mu xt^N(g))_{k,l} \cdot^M b_{l,j} \\
&= \sum_{k,l \leq n} a_{i,k} \cdot^M (\mu xt)'^M_{k,l}(\hat{g}) \cdot^M b_{l,j} \\
&= \sum_{k,l \leq n} \sum\{a_{i,k} \cdot^M (mxt)'^M_{k,l}(\hat{g}) \cdot^M b_{l,j} \mid m \in \mathbb{N}\} \quad \text{(Corollary 23)} \\
&= \sum\{\sum_{k,l \leq n} a_{i,k} \cdot^M (mxt)'^M_{k,l}(\hat{g}) \cdot^M b_{l,j} \mid m \in \mathbb{N}\} \\
&= \sum\{\sum_{k,l \leq n} a_{i,k} \cdot^M ((mxt)^N(g))_{k,l} \cdot^M b_{l,j} \mid m \in \mathbb{N}\} \\
&= \sum\{(A \cdot mxt^N(g) \cdot B)_{i,j} \mid m \in \mathbb{N}\}
\end{aligned}$$

Hence, N also is μ-continuous. ◀

7 Open Questions

In analogy to $\mathcal{C}X^*$, we can define the semiring of context-free subsets $\mathcal{C}M$ of an arbitrary monoid M by closing the semiring $\mathcal{F}M$ of finite subsets of M under least solutions in $\mathcal{P}M$ of polynomial inequations $\bar{x} \geq \bar{p}(\bar{x}, \bar{y})$ with parameters. Hopkins [5], [6] gives an elegant algebraic generalization of formal language theory, where he defines, for each monoid M, a dioid (= idempotent semiring) $\mathcal{C}M$ of context-free subsets of M in a different way. An open question is whether the two definitions of $\mathcal{C}M$ agree, and whether μ-continuous Chomsky algebras coincide with Hopkins' $\mathcal{C}$-dioids, just as *-continuous Kleene algebras coincide with his $\mathcal{R}$-dioids [6]. Hopkins left open whether the class of $\mathcal{C}$-dioids is closed under the matrix ring construction.

Another open question is whether $\mathcal{C}M$ can be constructed as an ideal-closure for a suitable notion of $\mathcal{C}$- or μ-ideal of M, if M is an idempotent semiring or a Kleene algebra.

Acknowledgements. I thank Mark Hopkins for comments and questions on drafts of this paper, for providing unpublished material of his extending [5], [6], and for email discussions on the subject. Thanks also to Dexter Kozen for making me aware of [11] and reviving my interest in the subject.

References

1 Hans Bekić. Definable operations in general algebras, and the theory of automata and flowcharts. In C. B. Jones, editor, *Programming Languages and Their Definition*, volume 177 of *LNCS*, pages 30–55. Springer-Verlag, Berlin Heidelberg, 1984.

2 J. H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.

3 J.W. de Bakker and D. Scott. A theory of programs. IBM Seminar, Vienna, 1969.

4 J. Gruska. A characterization of context-free languages. *Journal of Computer and System Sciences*, 5:353–364, 1971.

5 Mark Hopkins. The algebraic approach I: The algebraization of the Chomsky hierarchy. In *10th Int. Conf. on Relational Methods in Computer Science and 5th Int. Conf. on Applications of Kleene Algebra*, volume 4988 of *LNCS*, pages 155–172. Springer-Verlag, Berlin Heidelberg, 2008.

6 Mark Hopkins. The algebraic approach II: The algebraization of the Chomsky hierarchy. In *10th Int. Conf. on Relational Methods in Computer Science and 5th Int. Conf. on Applications of Kleene Algebra*, volume 4988 of *LNCS*, pages 173–190. Springer-Verlag, Berlin Heidelberg, 2008.

7 Dexter Kozen. Results in the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–345, 1983.

8 Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *6th Annual Symposium on Logic in Computer Science, July 15–18, 1991, Amsterdam*, Los Alamitos, CA, 1991. Computer Society Press.

9 H. Leiß. Towards Kleene Algebra with Recursion. In E. Börger, G. Jäger, H. Kleine-Büning, and M. M. Richter, editors, *CSL'91. 5th Workshop on Computer Science Logic. Berne, Switzerland, October 7–11, 1991*, pages 242–256. Springer LNCS 626, 1991.

10 Hans Leiß and Zoltan Ésik. Algebraically complete semirings and Greibach normal form. *Annals of Pure and Applied Logic*, 133:173–203, 2005.

11 Niels Bjørn Bugge Grathwohl, Fritz Henglein, and Dexter Kozen. Infinitary axiomatization of the equational theory of context-free languages. In D. Baelde and A. Carayol, editors, *Fixed Points in Computer Science (FICS 2013)*, volume 126 of *EPTCS*, pages 44–55, 2013. doi:10.4204/EPTCS.126.4.

A Appendix

Proof of claims following the definition of Park μ-semiring. For $t[x/\mu xt] = \mu xt$, we only need the $\geq$-inequation:

$$\begin{array}{rrcll} & t[x/\mu xt] & \leq & \mu xt & \text{(by (1))} \\ \Rightarrow & t[x/t[x/\mu xt]] & \leq & t[x/\mu xt] & \text{(by monotonicity)} \\ \Rightarrow & \mu xt & \leq & t[x/\mu xt] & \text{(by (2)).} \end{array}$$

For $\mu y.t[x/y] = \mu xt$ with $y \notin \mathit{free}(t)$,

$$\begin{array}{rrcll} & t[x/\mu y.t[x/y]] & = & t[x/y][y/\mu y.t[x/y]] & \\ & & \leq & \mu y.t[x/y] & \text{(by (1))} \\ \Rightarrow & \mu xt & \leq & \mu y.t[x/y] & \text{(by (2)).} \end{array}$$

By symmetry, we have $\mu y.t[x/y] \leq \mu x.t[x/y][y/x] \equiv \mu xt$. ◀

Proof of Lemma 8, properties of term functions. It remains to be seen that the properties of term functions in partially ordered μ-semirings hold in M for terms of μ-depth $\leq k$. We consider the substitution property. Suppose $r[x/s]$ has μ-depth $\leq k$, and the substitution property holds for terms of μ-depth $< k$. If x does not occur freely in r, then $r[x/s] = r$ and g agrees with $g[x/s^M(g)]$ on $\mathit{free}(r)$, hence $r[x/s]^M(g) = r^M(g) = r^M(g[x/s^M(g)])$. Otherwise, if r is one of $0, 1, y, (r_1 + r_2), (r_1 \cdot r_2)$, the claim holds by induction. So suppose $r = \mu yt$ with x different from y. We may assume that $y \notin \mathit{free}(s)$. Then $r[x/s] = \mu y.t[x/s]$, and by

induction, $t[x/s]^M(h) = t^M(h[x/s^M(h)])$ for all $h : X \to M$. Therefore, the least $a \in M$ with $t[x/s]^M(h[y/a]) \leq a$, is the least a with

$$t^M(h[x/s^M(h)][y/a]) = t^M(h[y/a][x/s^M(h[y/a])]) = t[x/s]^M(h[y/a]) \leq a,$$

which is $\mu y t^M(h[x/s^M(h)])$. Hence, $r[x/s]^M(h) = r^M(h[x/s^M(h)])$.

The other properties of term functions are shown similarly. Since M is an idempotent semiring, $+$ and $\cdot$ are monotone, hence t^M is monotone for any algebraic term t. By induction on the μ-depth, $\mu x t^M$ is monotone, and if $s^M \leq t^M$, we get $\mu x s^M \leq \mu x t^M$, since any a with $t^M(g[x/a]) \leq a$ satisfies $s^M(g[x/a]) \leq a$. It is also clear that $t^M(g) = t^M(h)$ if g agrees with h on $\mathit{free}(t)$. Hence, M is a partially ordered μ-semiring. ◀

Proof of Lemma 13. By induction on $|\bar{x}|$. For $|\bar{x}| = 1$, we have $\mu x t[\bar{y}/\bar{s}] = \mu x.t[\bar{y}/\bar{s}]$ by the definition of $[\bar{y}/\bar{s}]$. For $|\bar{x}| > 1$, assume $\bar{x} = (y, z)$, $\bar{t} = (r, s)$, and write $\bar{u}$ for $\bar{y}$, $\bar{v}$ for $\bar{s}$. Then, since $y, z \notin \mathit{free}(u)$,

$$\begin{aligned}
\mu\bar{x}\bar{t}[\bar{v}/\bar{u}] &= \mu(y,z)(r,s)[\bar{v}/\bar{u}] \\
&= (\mu y.r[z/\mu z s], \mu z.s[y/\mu y r])[\bar{v}/\bar{u}] \\
&= ((\mu y.r[z/\mu z s])[\bar{v}/\bar{u}], (\mu z.s[y/\mu y r])[\bar{v}/\bar{u}]) \\
&= (\mu y.r[z/\mu z s][\bar{v}/\bar{u}], \mu z.s[y/\mu y r][\bar{v}/\bar{u}]) \\
&= (\mu y.r[\bar{v}/\bar{u}][z/\mu z s[\bar{v}/\bar{u}]], \mu z.s[\bar{v}/\bar{u}][y/\mu y r[\bar{v}/\bar{u}]]) \\
&= (\mu y.r[\bar{v}/\bar{u}][z/\mu z.s[\bar{v}/\bar{u}]], \mu z.s[\bar{v}/\bar{u}][y/\mu y.r[\bar{v}/\bar{u}]]) \\
&= \mu(y,z)(r[\bar{v}/\bar{u}], s[\bar{v}/\bar{u}]) \\
&= \mu\bar{x}.\bar{t}[\bar{v}/\bar{u}].
\end{aligned}$$ ◀

Proof of Lemma 14. By induction on the dimension $|\bar{t}|$, we prove (10), (9) and, moreover, the substitution property for term vectors, for any $g : X \to M$, and $\bar{s}$ such that $\bar{x} \notin \mathit{free}(\bar{s})$,

$$\mu\bar{x}\bar{t}[\bar{y}/\bar{s}]^M(g) = \mu\bar{x}\bar{t}^M(g[\bar{y}/\bar{s}^M(g)]).$$

We suppress parameters in the notation and consider $\bar{x} = (\bar{y}, \bar{z})$, $\bar{t} = (\bar{r}(\bar{y}, \bar{z}), \bar{s}(\bar{y}, \bar{z}))$ with $|\bar{y}|, |\bar{z}| < |\bar{x}|$. For (10), suppose tuples $\bar{a}, \bar{b} \in A$ satisfy $(\bar{r}, \bar{s})^M(\bar{a}, \bar{b}) \leq (\bar{a}, \bar{b})$. By induction,

$$\mu\bar{y}\bar{r}^M[\bar{z}/\bar{b}] \leq \bar{a} \quad \text{and} \quad \mu\bar{z}\bar{s}^M[\bar{y}/\bar{a}] \leq \bar{b}.$$

By the substitution property and monotonicity of term vectors of dimension $< |\bar{t}|$ via (11),

$$\begin{aligned}
\bar{r}[\bar{z}/\mu\bar{z}\bar{s}]^M[\bar{y}/\bar{a}] &= \bar{r}^M[\bar{y}/\bar{a}, \bar{z}/\mu\bar{z}\bar{s}^M[\bar{y}/\bar{a}]) \\
&\leq \bar{r}^M[\bar{y}/\bar{a}, \bar{z}/\bar{b}] \\
&\leq \bar{a}.
\end{aligned}$$

Using (10) inductively, $\mu\bar{y}.\bar{r}[\bar{z}/\mu\bar{z}\bar{s}]^M \leq \bar{a}$. Likewise, $\mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]^M \leq \bar{b}$, and so we obtain

$$\mu(\bar{y}, \bar{z})(\bar{r}, \bar{s})^M = (\mu\bar{y}.\bar{r}[\bar{z}/\mu\bar{z}\bar{s}]^M, \mu\bar{z}.\bar{s}[\bar{y}/\mu\bar{y}\bar{r}]^M) \leq (\bar{a}, \bar{b}).$$

To show (9), we improve readability by writing substitutions in place, i.e.

$$\bar{t}[\bar{x}/\mu\bar{x}\bar{t}] = (\bar{r}(\mu\bar{y}.\bar{r}(\bar{y}, \mu\bar{z}\bar{s}), \mu\bar{z}.\bar{s}(\mu\bar{y}\bar{r}, \bar{z})), \bar{s}(\mu\bar{y}.\bar{r}(\bar{y}, \mu\bar{z}\bar{s}), \mu\bar{z}.\bar{s}(\mu\bar{y}\bar{r}, \bar{z}))).$$

As both components of $\bar{t}[\bar{x}/\mu\bar{x}\bar{t}]^M \leq \mu\bar{x}\bar{t}^M$ can be treated alike, we only show the first one,

$$\bar{r}(\mu\bar{y}.\bar{r}(\bar{y}, \mu\bar{z}\bar{s}), \mu\bar{z}.\bar{s}(\mu\bar{y}\bar{r}, \bar{z}))^M \leq \mu\bar{y}.\bar{r}(\bar{y}, \mu\bar{z}\bar{s})^M. \tag{24}$$

By induction, $\bar{r}(\bar{y},\mu\bar{z}\bar{s})[\bar{y}/\mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s})]^M \le \mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s})^M$, so for (24) it is sufficient to prove

$$\bar{r}(\mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s}),\mu\bar{z}.\bar{s}(\mu\bar{y}\bar{r},\bar{z}))^M \le \bar{r}(\bar{y},\mu\bar{z}\bar{s})[\bar{y}/\mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s})]^M,$$

for which, by monotonicity of $\bar{r}^M$, it's sufficient to show

$$\mu\bar{z}.\bar{s}(\mu\bar{y}\bar{r},\bar{z})^M \le \mu\bar{z}.\bar{s}(\mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s}),z)^M. \tag{25}$$

Let $\bar{a} = \mu\bar{y}.\bar{r}(\bar{y},\mu\bar{z}\bar{s})^M$ and $\bar{b}$ the least $\bar{b}'$ with $\bar{s}^M(\bar{a},\bar{b}') \le \bar{b}'$, i.e. $\bar{b} = \mu\bar{z}\bar{s}^M[\bar{y}/\bar{a}]$. Then

$$\begin{array}{lcll} \bar{r}^M(\bar{a},\bar{b}) & = & \bar{r}[z/\mu\bar{z}\bar{s}]^M(\bar{a}) & \text{(by (11), inductively)} \\ & = & \bar{r}(\bar{y},\mu\bar{z}\bar{s})^M[y/\bar{a}] & \\ & \le & \bar{a} & \text{(by (9), (11), inductively),} \end{array}$$

hence $\mu\bar{y}\bar{r}^M[z/b] \le \bar{a}$ by an application of (10). By monotonicity, it follows that

$$\begin{array}{lcll} \bar{s}(\mu\bar{y}.\bar{r}(\bar{y},\bar{z}),\bar{z})^M[\bar{z}/\bar{b}] & \le & \bar{s}^M(\bar{a},\bar{b}) & \\ & \le & \bar{b} & \text{(by the choice of } \bar{b}). \end{array}$$

An application of (10) to this gives $\mu\bar{z}.\bar{s}(\mu\bar{y}.\bar{r}(\bar{y},\bar{z}),\bar{z})^M \le \bar{b}$, which is (25).

To show (11) for $|\bar{t}| > 1$, by induction it follows from Lemma 13 that

$$\begin{array}{lcl} \mu\bar{x}\bar{t}[\bar{y}/\bar{s}]^M(g) & = & (\mu\bar{x}.\bar{t}[\bar{y}/\bar{s}])^M(g) \\ & = & \text{the least } \bar{a} \text{ such that } \bar{t}[\bar{y}/\bar{s}]^M(g[\bar{x}/\bar{a}]) \le \bar{a} \\ & = & \text{the least } \bar{a} \text{ such that } \bar{t}^M(g[\bar{x}/\bar{a}][\bar{y}/\bar{s}^M(g[\bar{x}/\bar{a}])]) \le \bar{a} \\ & = & \text{the least } \bar{a} \text{ such that } \bar{t}^M(g[\bar{y}/\bar{s}^M(g)][\bar{x}/\bar{a}]) \le \bar{a} \\ & = & \mu\bar{x}\bar{t}^M(g[\bar{y}/\bar{s}^M(g)]). \end{array}$$

The case $|\bar{t}| = 1$ is an instance of the substitution property for M. ◀

Omitted bits in the proof of Theorem 25. The properties of 0^N, 1^N, x^N, $(s+t)^N$ and $(s\cdot t)^N$ are immediate from the definition. For example,

$$(s+t)^N(g) = ((s+t)'^M_{i,j}(\hat{g})) = (s'^M_{i,j}(\hat{g}) +^M t'^M_{i,j}(\hat{g})) = s'^N(g) + t'^N(g).$$

The μ-rule holds in N, since, by corollary 15, the vector version of the μ-rule holds in M. Namely, suppose $s^N \le t^N$, and $g : X \to N$. Then $\mu x t^N(g) = \mu x' t'^M(\hat{g})$ is the least $\bar{a}$ such that $t'^M(\hat{g}[x'/\bar{a}]) \le \bar{a}$ by lemma 14. From $s^N \le t^N$ we get $s'^M(\hat{g}[x'/\bar{a}]) \le t'^M(\hat{g}[x'/\bar{a}]) \le \bar{a}$, which implies $\mu x s^N(g) \le \mu x t^N(g)$. ◀

Completeness for Coalgebraic Fixpoint Logic

Sebastian Enqvist[1], Fatemeh Seifan[2], and Yde Venema[3]

1 **ILLC, Universiteit van Amsterdam, The Netherlands; and Department of Philosophy, Lund University, Sweden**
`sebastian.enqvist@fil.lu.se`
2 **ILLC, Universiteit van Amsterdam, The Netherlands**
`F.Seifan@uva.nl`
3 **ILLC, Universiteit van Amsterdam, The Netherlands**
`Y.Venema@uva.nl`

Abstract

We introduce an axiomatization for the coalgebraic fixed point logic which was introduced by Venema as a generalization, based on Moss' coalgebraic modality, of the well-known modal mu-calculus. Our axiomatization can be seen as a generalization of Kozen's proof system for the modal mu-calculus to the coalgebraic level of generality. It consists of a complete axiomatization for Moss' modality, extended with Kozen's axiom and rule for the fixpoint operators. Our main result is a completeness theorem stating that, for functors that preserve weak pullbacks and restrict to finite sets, our axiomatization is sound and complete for the standard interpretation of the language in coalgebraic models. Our proof is based on automata-theoretic ideas: in particular, we introduce the notion of consequence game for modal automata, which plays a crucial role in the proof of our main result. The result generalizes the celebrated Kozen-Walukiewicz completeness theorem for the modal mu-calculus, and our automata-theoretic methods simplify parts of Walukiewicz' proof.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases μ-calculus, coalgebra, coalgebraic modal logic, automata, completeness

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.7

1 Introduction

The modal μ-calculus ($\mu\mathtt{ML}$) was introduced Kozen [7] as a logic obtained by adding least and greatest fixpoint operators to modal logic. It is of great interest to computer science for expressing properties of processes such as termination and fairness. The power of the μ-calculus is also evident from a more theoretical perspective: adding fixpoint operators significantly increases the expressiveness of the formalism. In particular well-known temporal logics like LTL, CTL and CTL* can be defined in terms of the μ-calculus. A key result concerning the expressive strength of the μ-calculus is the Janin-Walukiewicz theorem [5], which states that a formula φ of monadic second-order logic ($\mathtt{MSO}$) is equivalent to a μ-formula iff φ is invariant under bisimulation. Thus the μ-calculus seems a well-suited specification language, but there is a drawback: the μ-calculus is a complex formalism to work with.

A specific manifestation of this complexity lies in the axiomatization problem. In the same paper where he introduced the μ-calculus [7], Kozen also suggested an axiomatization and proved that his axiomatization is complete for a fragment of $\mu\mathtt{ML}$, consisting of the so-called *aconjunctive* formulas. The completeness of Kozen's axiomatization for the full language remained an open problem for many years, but eventually, Walukiewicz [18] provided a positive answer to this question. Regrettably his landmark result has remained a stand-alone

© Sebastian Enqvist, Fatemeh Seifan, and Yde Venema;
licensed under Creative Commons License CC-BY

theorem in the theory of (fixpoint) modal logic. This is largely due to the complexity of Walukiewicz' proof, which is based on an intricate mix of ideas from automata theory, game theory and logic (such as tableaux). It is the aim of our paper to clarify and generalize Walukiewicz' proof by focusing on ideas from automata theory and coalgebra.

Automata-theoretic methods lie on the heart of the theory of modal μ-calculus. In fact, most of the model-theoretic results on $\mu\mathtt{ML}$, such as D'Agostino and Hollenberg's characterization and uniform interpolation theorems [2] and the Janin-Walukiewicz theorem mentioned earlier, are based on Walukiewicz' result that formulas of $\mu\mathtt{ML}$ effectively correspond to what we shall call *disjunctive* automata: non-deterministic parity automata operating on Kripke models. This approach builds on a long tradition connecting logic and automata theory, going back to the seminal work of Büchi, Rabin and others. As examples we mention Rabins' decidability theorem [14], and the result by Büchi [1] showing that finite automata and monadic second order logic ($\mathtt{MSO}$) have the same expressive power over infinite words.

Coalgebra enters this framework in a natural way. Recall from [15] that coalgebra uniformly generalizes state-based evolving systems such as streams, (infinite) trees, Kripke models, transition systems, and many others, by encoding the type of a dynamic system into a functor T. Starting with Moss' seminal paper [12], coalgebraic logics have been developed, and with Kripke structures constituting key examples of coalgebras, it should come as no surprise that most coalgebraic logics are some kind of modification or generalization of modal logic. Interestingly, Moss' logic centers around a coalgebraic generalization of the same cover modality ∇ that is the main operator featuring in Walukiewicz' completeness proof. Extending Moss' logic with fixpoint operators, Venema [17] introduced the coalgebraic fixpoint logic $\mu\mathtt{ML}_{\mathsf{T}}$, where T denotes the set functor encoding the coalgebra type. In the same paper Venema proved the existence of effective meaning preserving translations between formulas of $\mu\mathtt{ML}_{\mathsf{T}}$ and coalgebraic modal automata.

Here we address the completeness question for this coalgebraic fixpoint logic $\mu\mathtt{ML}_{\mathsf{T}}$. Note that a complete axiomatisation $\mathbf{M}$ for basic coalgebraic modal logic, i.e., the logic $\mu\mathtt{ML}_{\mathsf{T}}$ without the fixpoint operators, was given in [9]. Hence another way to view the result of this paper is that it extends the completeness result of [9] to the setting of fixpoint logic. In particular, our axiom system $\mathbf{K}$ is an extension of the system $\mathbf{M}$ with Kozen's axiom and derivation rule.

Just as in Walukiewicz' proof, we use translations between formulas of $\mu\mathtt{ML}_{\mathsf{T}}$ and automata. The difference is that we will be more radical and bring automata into the picture at an earlier stage. The main goal in Walukiewicz' proof strategy is to show that every formula of the μ-calculus *provably* implies a semantically equivalent disjunctive formula, that is, a modal μ-formula in a normal form corresponding to the disjunctive automata mentioned earlier. Here, we shall work with the full class of coalgebraic modal automata, for which an analogous result can be proved by much more elementary techniques. More specifically, our proof is based on translations between formulas of $\mu\mathtt{ML}_{\mathsf{T}}$ and coalgebraic automata (respectively denoted by $\mathbb{A}_\varphi$ for a formula $\varphi \in \mu\mathtt{ML}_{\mathsf{T}}$ and $\mathsf{tr}(\mathbb{A}) \in \mu\mathtt{ML}_{\mathsf{T}}$ for an automaton $\mathbb{A}$) for which we have the following proposition (with $\equiv_{\mathbf{K}}$ denoting provable equivalence with respect to system $\mathbf{K}$).

▶ **Proposition.** *For every formula $\varphi \in \mu\mathtt{ML}_{\mathsf{T}}$, we have $\varphi \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A}_\varphi)$.*

This proposition takes us *half-way* towards Walukiewicz' result, and we address the remaining half of the distance by automata-theoretic techniques. In this way we can make the key concept of a *trace*, which is an essential but fairly informally discussed ingredient in Walukiewicz' proof, more explicit by developing a framework for 'managing' traces. Thus our machinery separates *dynamics* (coalgebra) from *combinatorics* (trace management) and

this simplifies the proof in two ways. First, the coalgebraic perspective on automata uses a strictly controlled syntax and semantics via the so-called *one-step* framework, and second, it allows us to handle traces more explicitly. More in detail, our approach focus on two automata-related games: we will work with the *satisfiability game* of [3], which comprises a streamlined analogue of the logical notion of tableau, and we introduce the *consequence game* between two automata. Informally, the latter game, which resembles Walukiewicz' consequence game between tableaux, can be seen as a kind of implication game between the satisfiability games of two automata, revolving around establishing *structural connections* between the automata.

Making traces first-class citizens in our approach and bringing the trace theory to the surface, we will arrive at isolating classes of automata which allow a relatively simple trace management. The first automata that naturally appear are, once again, the *disjunctive automata*. These automata admit a *trivial* trace theory, in the sense that the matches of the satisfiability game of a disjunctive automaton involves a single trace only.

Bringing all these ideas together, as a key step in our completeness proof, we prove the following generalization of one of Walukiewicz' lemmas:

▶ **Theorem 1.** *For every formula $\varphi \in \mu\mathtt{ML}_\mathsf{T}$, there is a semantically equivalent disjunctive automaton $\mathbb{D}$ such that $\vdash_{\mathbf{K}} \varphi \to \mathsf{tr}(\mathbb{D})$.*

The proof of this theorem is by induction on the complexity of φ. In order to handle the induction steps we need to introduce a second class of special automata, namely *semi-disjunctive automata*, which roughly correspond to the *aconjunctive* formulas in Kozen's proof and the *weakly aconjunctive* formulas in the Walukiewicz' proof. These automata are more general than the disjunctive ones but still have a relatively *simple* (though not trivial) trace theory: the collection of *bad traces* in any match in the satisfiability game of a semi-disjunctive automaton is essentially finite. Finally, the completeness theorem itself is fairly straightforward corollary of Theorem 1.

Finally, we should mention that the completeness proof we provide here is still long and full of technical details, and for this reason we present this paper as an extended abstract. We hope that our 'deconstruction' of Walukiewicz' proof will contribute to a better understanding of the completeness theory of fixpoint logics. As a sample, in future work we will generalize this result to a wider coalgebraic context. As a first step we will use the results reported on here to provide a completeness result for *monotone fixpoint modal logic*.

Overview. We first fix notation and terminology on Set-based functors and coalgebras. In section 3 we recall the definition of the coalgebraic fixpoint logic $\mu\mathtt{ML}_\mathsf{T}$ and we introduce the proof system $\mathbf{K}$. Section 4 is concerned with the definition of the one-step framework and the coalgebraic modal automata corresponding to formulas of $\mu\mathtt{ML}_\mathsf{T}$. In this section we also define the satisfiability and consequence games for modal automata. Section 5 is devoted to the introduction of the disjunctive and semi-disjunctive automata. We discuss some of the closure properties of these automata and we state two of the main results of our paper, viz., Theorem 38 and Theorem 39. Finally, in section 6 we combine the results from the sections 4 and 5 in order to prove the completeness of the system $\mathbf{K}$ for $\mu\mathtt{ML}_\mathsf{T}$.

2 Preliminaries

In this section we settle on notation and terminology. For background on coalgebra the reader is referred to [15].

General. In this paper we work with two categories: the category Set with sets as objects and functions between sets as arrows. The composition of two functions $f : X \to Y$ and $g : Y \to Z$ is written as $g \circ f$. For a function $f : X \to Y$ and a set $X' \subseteq X$ we define the restriction of f to X' as $f{\restriction_{X'}} : X' \to Y$, $x \mapsto f(x)$. The next category is the category Rel with sets as objects and relations as arrows. The composition of relations $R : X \nrightarrow Y$ and $S : Y \nrightarrow Z$ is written as $R \,;\, S$. Given a function f we use the same symbol f to refer to the graph of f.

Coalgebra. A coalgebra (over Set) for a functor $\mathsf{T} : \mathsf{Set} \to \mathsf{Set}$, also called T-coalgebra, is a pair (X, σ) where S is a set (of states) and $\sigma : X \to \mathsf{T}X$ is a function (the transition structure). A T-coalgebra morphism from a T-coalgebra (X_1, σ_1) to a T-coalgebra (X_2, σ_2) is a function $f : X_1 \to X_2$ such that $\mathsf{T}f \circ \sigma_1 = \sigma_2 \circ f$. The prime examples of coalgebras for modal logicians are Kripke frames and Kripke models. Bisimulations between Kripke structures also have their natural coalgebraic generalization: a relation Z between the carrier sets of two coalgebras is a bisimulation if for all $(x_1, x_2) \in Z$, the pair $(\sigma_1(x_1), \sigma_2(x_2))$ belongs to the *lifting* $\overline{\mathsf{T}}Z$ of relation Z.

▶ **Definition 2.** Let T be a set functor. Given a binary relation Z between two sets X_1 and X_2, we define the relation $\overline{\mathsf{T}}Z : \mathsf{T}X_1 \nrightarrow \mathsf{T}X_2$ as follows:

$$\overline{\mathsf{T}}Z := \{((\mathsf{T}\pi_1)\rho, (\mathsf{T}\pi_2)\rho) \mid \rho \in \mathsf{T}Z\}$$

where $\pi_i : Z \to X_i$ for $i = 1, 2$ are the projection maps.

In this paper we will confine attention to set functors that preserve finite sets (that is, $\mathsf{T}X$ is a finite set if X is finite) and weak pullbacks. The latter property, which plays an important role in the theory of coalgebras, is defined as follows:

▶ **Definition 3.** A functor T preserves weak pullbacks if it maps every weak pullback (P, p_1, p_2) of maps $f_1 : X_1 \to Y$ and $f_2 : X_2 \to Y$ onto a weak pullback $(\mathsf{T}P, \mathsf{T}p_1, \mathsf{T}p_2)$ of $\mathsf{T}f_1 : \mathsf{T}X_1 \to \mathsf{T}Y$ and $\mathsf{T}f_2 : \mathsf{T}X_2 \to \mathsf{T}Y$.

▶ **Convention 4.** *Throughout this paper we fix a functor* T *that preserves weak pullbacks and finite sets. We also assume that* T *preserves inclusions and finite intersections, but this is without loss of generality (see Convention 2.7. of [11]).*

The following fact lists the properties of relation lifting $\overline{\mathsf{T}}$ that we use in our paper. For proofs we refer to [12] and references therein.

▶ **Fact 5.** *Let* T *be a set functor that preserves inclusions and weak pullbacks. Then relation lifting* $\overline{\mathsf{T}}$

1. *extends* T*:* $\overline{\mathsf{T}}f = \mathsf{T}f$*;*
2. *is monotone:* $R \subseteq Q$ *implies* $\overline{\mathsf{T}}R \subseteq \overline{\mathsf{T}}Q$*;*
3. *commutes with taking restrictions:* $\overline{\mathsf{T}}(R{\restriction_{X \times X'}}) = (\overline{\mathsf{T}}R){\restriction_{\mathsf{T}X \times \mathsf{T}X'}}$*;*
4. *preserves composition:* $\overline{\mathsf{T}}(R \,;\, Q) = \overline{\mathsf{T}}R \,;\, \overline{\mathsf{T}}Q$ *and converse:* $\overline{\mathsf{T}}(R^\circ) = (\overline{\mathsf{T}}R)^\circ$*.*

Lifting of special relations, like the membership relation, is used to define notions that will be used in the next section.

We first define the notion of *Base*.

▶ **Definition 6.** Given a functor T we define its subfunctor T_ω to be the functor that maps a set X to $\mathsf{T}_\omega X = \bigcup\{\mathsf{T}X' \mid X' \subseteq X,\ X' \text{ is finite }\}$. Then for every set X we define the function

$$Base_X : \mathsf{T}_\omega X \to \mathcal{P}_\omega X,\ \alpha \mapsto \bigcap\{X' \subseteq X \mid \alpha \in \mathsf{T}X'\}.$$

One may prove that $Base_X(\alpha)$ is the smallest set $U \in \mathcal{P}_\omega X$ such that $\alpha \in \mathsf{T}_\omega U$.

▶ **Definition 7.** Given a set X, we let $\in_X \subseteq X \times \mathcal{P}X$ denote the membership relation, restricted to X. We define the maps $\lambda^{\mathsf{T}}_X : \mathsf{T}\mathcal{P}X \to \mathcal{P}\mathsf{T}X$ by

$$\lambda^{\mathsf{T}}_X(\Phi) := \{\alpha \in \mathsf{T}X \mid \alpha \,\overline{\mathsf{T}}{\in_X}\, \Phi\}$$

and call members of $\lambda^{\mathsf{T}}_X(\Phi)$ *lifted members* of Φ. An object $\Phi \in \mathsf{T}\mathcal{P}X$ is a *redistribution of* $\Gamma \in \mathcal{P}\mathsf{T}X$ if $\Gamma \subseteq \lambda^{\mathsf{T}}_X(\Phi)$. In case $\Gamma \in \mathcal{P}_\omega\mathsf{T}_\omega X$, we call a redistribution Φ *slim* if $\Phi \in \mathsf{T}_\omega\mathcal{P}_\omega(\bigcup_{\alpha\in\Gamma} Base(\alpha))$. The set of all slim redistributions of Γ is denoted as $SRD(\Gamma)$.

Slim redistributions will be later used in order to define an axiom system for the coalgebraic fixpoint logic.

3 Coalgebraic fixpoint logic

3.1 Syntax and semantics

In this section we recall from [17] the syntax and semantics of a version of coalgebraic fixpoint logic which is based on Moss modality.

▶ **Definition 8.** We fix an infinite set of propositional variables. The language $\mu\mathtt{ML}_{\mathsf{T}}$ of coalgebraic fixpoint formulas is defined by the following grammar:

$$\varphi ::= \bot \mid \top \mid p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \nabla\alpha \mid \neg\varphi \mid \mu p.\varphi \mid \nu p.\varphi$$

where p belongs to the set of propositional variables, and $\alpha \in \mathsf{T}_\omega(\mu\mathtt{ML}_{\mathsf{T}})$. There is a restriction on the formation of the formulas $\mu p.\varphi$ and $\nu p.\varphi$, namely, no occurrence of p in φ may be in the scope of an odd number of negations. We denote by $\mu\mathtt{ML}_{\mathsf{T}}(\mathtt{X})$ the set of formulas with free variables from set $\mathtt{X}$, and as a convention we usually use letters $p, q, r, \ldots$ to denote bound variables and $x, y, z, \ldots$ for free variables of formulas.[1]

Readers who worry about the well-definedness of the inductive clause for ∇ may observe that since T_ω is a *finitary* functor, what we are saying is simply that for any finite set X of formulas, any object $\alpha \in \mathsf{T}_\omega(X)$ is a formula as well. In fact, any $\alpha \in \mathsf{T}_\omega(\mu\mathtt{ML}_{\mathsf{T}})$ belongs to the set $\mathsf{T}_\omega(Base(\alpha))$, and we will call the formulas in $Base(\alpha)$ the *immediate subformulas* of the formula $\nabla\alpha$.

To introduce the semantics of $\mu\mathtt{ML}_{\mathsf{T}}$ we first define the notion of a T-model over a set $\mathtt{X}$ of propositional letters.

▶ **Definition 9.** A T-model $\mathbb{S} = (S, \sigma, m)$ is a T-coalgebra (S, σ) together with a marking $m : S \to \mathcal{P}\mathtt{X}$. It will be convenient to think of a T-model $\mathbb{S}$ as a coalgebra $\mathbb{S} = (S, \sigma_m)$ for functor $\mathsf{T}_{\mathtt{X}}$ defined by $\mathsf{T}_{\mathtt{X}}S := \mathcal{P}\mathtt{X} \times \mathsf{T}S$ where $\sigma_m : S \to \mathsf{T}_{\mathtt{X}}S$ is given by map (m, σ). It is obvious that any marking $m : S \to \mathcal{P}\mathtt{X}$ induces a valuation $V_m : \mathtt{X} \to \mathcal{P}S$ mapping p to the set $\{s \in S \mid p \in m(s)\}$.

[1] For a precise definition of the notions *scope* and *occurrence*, we inductively define the construction tree of a formula, where the children of a node labeled $\nabla\alpha$ are the formulas in $Base(\alpha)$. For the definition of free and bound variables see [17].

Table 1 Rules and axioms of the system **K**.

$(\Delta 1)$	$\dfrac{\{\varphi \to \psi \mid (\varphi,\psi) \in Z\}}{\nabla\alpha \to \nabla\beta}(\alpha,\beta) \in \overline{\mathsf{T}}Z$
$(\Delta 2)$	$\bigwedge\{\nabla\alpha \mid \alpha \in \Gamma\} \to \bigvee\{\nabla(\mathsf{T}\wedge)(\Phi) \mid \Phi \in SRD(\Gamma)\}$
$(\Delta 3)$	$\nabla(\mathsf{T}\vee)(\Phi) \to \bigvee\{\nabla\alpha \mid \alpha\ \overline{\mathsf{T}}{\in}\ \Phi\}$
(A_f)	$\varphi(\mu x.\varphi(x)) \to \mu x.\varphi(x)$
(R_f)	$\dfrac{\varphi(\psi) \to \psi}{\mu x.\varphi \to \psi}$

Using the relation lifting $\overline{\mathsf{T}}$ we define the semantics for the language $\mu\mathtt{ML}_{\mathsf{T}}(\mathtt{X})$ on T-models. Since apart from the nabla modality, the definition of the satisfaction relation $\Vdash_{V_m}$ is exactly the same as it is for the μ-calculus, here we only give the definition for the nabla modality[2]:

$$s \Vdash_{V_m} \nabla\alpha \text{ iff } (\sigma_m(s),\alpha) \in \overline{\mathsf{T}}(\Vdash_{V_m}).$$

See [11] for more details.

The standard modal μ-calculus is the logic $\mu\mathtt{ML}_{\mathcal{P}}$ for the power set functor $\mathcal{P}$. In this case ∇ is known as the *cover modality*. It can be expressed using $\Box$ and $\Diamond$:

$$\nabla A = \Box\bigvee A \wedge \bigwedge \Diamond A;$$

where $\Diamond A$ denotes the set $\{\Diamond a \mid a \in A\}$. Conversely we have:

$$\Diamond a = \nabla\{a,\top\} \quad \text{and} \quad \Box a = \nabla\emptyset \vee \{a\}.$$

It is not difficult to see that in this case $\mathcal{P}_{\mathtt{X}}$-coalgebras are standard Kripke models.

3.2 Derivation system K

Our derivation system **K** is the extension of the complete derivation system **M** for Moss' finitary logic [9][8] with rules and axioms for the fixpoint operators.

▶ **Definition 10.** The derivation system **K** which is uniformly parametric in the functor T is given by the following axioms and derivation rules, together with any complete set of axioms and rules for classical propositional logic. The rules and axioms of **K** are given in Table 1.

The axioms $(\Delta 2)$ and $(\Delta 3)$ are governing the interaction of ∇ with conjunctions and disjunctions respectively and can be seen as *modal distributive laws*. Here we see conjunction and disjunction ($\wedge$ and $\vee$) as maps from $\mathcal{P}_\omega(\mu\mathtt{ML}_{\mathsf{T}})$ to $\mu\mathtt{ML}_{\mathsf{T}}$, so we can apply T to them and get maps $\mathsf{T}\wedge$ and $\mathsf{T}\vee$. The rule $(\Delta 1)$ can be read as a congruence and monotonicity rule in one. It has a side condition expressing that it may only be applied when the set of premisses is indexed by a relation Z such that (α,β) belongs to the lifted relation $\overline{\mathsf{T}}Z$. (A_f) and (R_f) are the standard axiom and rule for pre-fixpoint.

[2] Strictly speaking the clause for nabla in Definition 9 is not stated in a correct recursive way, since it makes use of the unrestricted satisfaction relation $\Vdash_{V_m}$ that has yet to be defined. We can only suppose that $\Vdash_{V_m}|_{S\times Base(\alpha)}$ is already defined. The actual recursive definition is that $s \Vdash_{V_m} \nabla\alpha$ iff $(\sigma(s),\alpha) \in \overline{\mathsf{T}}(\Vdash_{V_m}|_{S\times Base(\alpha)})$. One can apply Fact 5 item (3) to prove that this definition is equal to the clause given above.

The notions of *derivability* with respect to this system is standard. If there is a derivation of the formula φ we write $\vdash_{\mathbf{K}} \varphi$. We write $\varphi \equiv_{\mathbf{K}} \psi$ in the case that both $\vdash_{\mathbf{K}} \varphi \to \psi$ and $\vdash_{\mathbf{K}} \psi \to \varphi$ hold. A formula φ is $\mathbf{K}$-*consistent* or simply *consistent* if $\varphi \to \bot$ is not derivable in $\mathbf{K}$.

▶ **Example 11.** In the case of the power set functor ($\mathsf{T} = \mathcal{P}$) the axioms ($\Delta 2$) and ($\Delta 3$) look as follows:

$$\bigwedge\{\nabla\alpha \mid \alpha \in \Gamma\} \to \bigvee\{\nabla\{\bigwedge\beta \mid \beta \in \Phi\} \mid \bigcup\Gamma = \bigcup\Phi \text{ and } \alpha \cap \beta \neq \emptyset \text{ for all } \alpha \in \Gamma, \beta \in \Phi\}$$

$$\nabla\{\bigvee\beta \mid \beta \in \Phi\} \to \bigvee\{\nabla\alpha \mid \alpha \subseteq \bigcup\Phi \text{ and } \alpha \cap \beta \neq \emptyset \text{ for all } \beta \in \Phi\}$$

4 Coalgebraic modal automata

In this section, we lift the well-known relation between automata and logic to a coalgebraic level. We first recall the notions of one-step syntax and semantics.

4.1 One-step logic

Given a set A, we define the set $\mathtt{Latt}(A)$ of lattice terms over A through the following grammar:

$$\pi ::= \bot \mid \top \mid a \mid \pi \wedge \pi \mid \pi \vee \pi,$$

where $a \in A$. Given two sets $\mathtt{X}$ and A, we define the set $\mathtt{1ML}_{\mathsf{T}}(\mathtt{X}, A)$ of *modal one-step formulas* over A with respect to $\mathtt{X}$ inductively by

$$\alpha ::= \bot \mid \top \mid p \mid \neg p \mid \nabla\beta \mid \alpha \wedge \alpha \mid \alpha \vee \alpha,$$

with $p \in \mathtt{X}$ and $\beta \in \mathsf{T}_\omega\mathtt{Latt}(A)$.

Any valuation $V : A \to \mathcal{P}S$ can be extended to a meaning function $[\![-]\!]^0_V : \mathtt{Latt}(A) \to \mathcal{P}S$ in the usual way. We write $S, s \Vdash^0_V \varphi$ to indicate $s \in [\![\varphi]\!]^0_V$. The meaning function $[\![-]\!]^0_V$ induces a map $[\![-]\!]^1_V : \mathtt{1ML}_{\mathsf{T}}(\mathtt{X}, A) \to \mathcal{P}\mathsf{T}_{\mathtt{X}}S$ interpreting one-step formulas as subsets of $\mathsf{T}_{\mathtt{X}}S$. Before giving the definition of $[\![-]\!]^1_V$ we recall that every $\tau \in \mathsf{T}_{\mathtt{X}}S$ is of the form $(\mathtt{Y}, \tau') \in \mathcal{P}\mathtt{X} \times \mathsf{T}S$.

Going back to the map $[\![-]\!]^1_V$, it has the usual clauses for conjunction and disjunction, and the following clauses for the propositional letters and the modal operator:

- $\tau = (\mathtt{Y}, \tau') \in [\![p]\!]^1_V$ iff $p \in \mathtt{Y}$
- $\tau = (\mathtt{Y}, \tau') \in [\![\neg p]\!]^1_V$ iff $p \notin \mathtt{Y}$
- $\tau = (\mathtt{Y}, \tau') \in [\![\nabla\beta]\!]^1_V$ iff $(\tau', \beta) \in \overline{\mathsf{T}}(\Vdash^0_V)$

We write $S, \tau \Vdash^1_V \varphi$ to indicate $\tau \in [\![\varphi]\!]^1_V$, and refer to this relation as the *one-step semantics*.

For technical reasons, we need the following *binary* version of the modal distributive law for one-step conjunctions.

▶ **Proposition 12.** *Given $\alpha_1, \alpha_2 \in \mathsf{T}_\omega\mathtt{Latt}(A)$ the following holds:*

$$\nabla\alpha_1 \wedge \nabla\alpha_2 \equiv_K \bigvee\{\nabla(\mathsf{T}\wedge)\alpha \mid \alpha \in \mathsf{T}(\mathit{Base}(\alpha_1) \times \mathit{Base}(\alpha_2)) \text{ and } \mathsf{T}\pi_i(\alpha) = \alpha_i \text{ for } i \in \{1, 2\}\}$$

where the conjunction on the right hand side is a map $\wedge : \mathtt{Latt}(A) \times \mathtt{Latt}(A) \to \mathtt{Latt}(A)$.

To prove this result one can use properties of weak pullback preserving functors and show that these formulas are semantically equivalent and then from the one-step completeness result of [9] derive that they are provably equivalent.

Table 2 Acceptance Game.

Position	Player	Admissible moves
$(a, s) \in A \times S$	$\exists$	$\{V : A \to \mathcal{P}S \mid \sigma_m(s) \in [\![\Theta(a)]\!]^1_V\}$
V	$\forall$	$\{(b, t) \in A \times S \mid t \in V(b)\}$

4.2 Modal automata

▶ **Definition 13.** A *(coalgebraic) modal* $\mathtt{X}$*-automaton* is a quadruple $\mathbb{A} = (A, \Theta, \Omega, a_I)$ such that A is a finite set of *states*, $\Theta : A \to \mathtt{1ML_T}(\mathtt{X}, A)$ is the *transition map* of $\mathbb{A}$, $\mathtt{X}$ is the set of free variables of formulas in the range of the transition map Θ, $\Omega : A \to \omega$ is the *priority map* of $\mathbb{A}$ and a_I is the *initial state*. We define the notions of free and positive occurrences of x in $\mathbb{A}$ in the obvious way.

▶ **Definition 14.** The *(directed) graph* of $\mathbb{A}$ is the structure $(A, E_{\mathbb{A}})$, where $aE_{\mathbb{A}}b$ if a occurs in $\Theta(b)$, and we let $\lhd_{\mathbb{A}}$ denote the transitive closure of $E_{\mathbb{A}}$. If $a \lhd_{\mathbb{A}} b$ we say that a is *active* in b. We write $a \bowtie_{\mathbb{A}} b$ if $a \lhd_{\mathbb{A}} b$ and $b \lhd_{\mathbb{A}} a$. A *cluster* of $\mathbb{A}$ is a cell of the equivalence relation generated by $\bowtie_{\mathbb{A}}$; a cluster C is *degenerate* if it is of the form $C = \{a\}$ with $a \not\bowtie_{\mathbb{A}} a$. Given a state a of $\mathbb{A}$, we write $\eta_a = \mu$ if $\Omega(a)$ is odd, and $\eta_a = \nu$ if $\Omega(a)$ is even, and we call state a a η_a-state. The sets of μ- and ν-states are denoted with A^μ and A^ν, respectively. For a state a we denote by C_a the unique cluster of $\mathbb{A}$ to which a belongs.

▶ **Definition 15.** Let $\mathbb{A} = (A, \Theta, \Omega, a_I)$ be a modal $\mathtt{X}$-automaton and let $\mathbb{S} = (S, \sigma, m)$ be a T-model. The associated acceptance game $\mathcal{A}(\mathbb{A}, \mathbb{S})$ is the parity game given by Table 2.

The loser of a finite match is the player who got stuck; the winner of an infinite match is $\exists$ if the greatest parity that appears infinitely often in the match is even, and it is $\forall$ if this parity is odd. A pointed coalgebra $(\mathbb{S}, s_I)$ is *accepted* by the automaton $\mathbb{A}$ if (a_I, s_I) is a winning position for player $\exists$ in $\mathcal{A}(\mathbb{A}, \mathbb{S})$. We denote the *language recognized by* $\mathbb{A}$ with $L(\mathbb{A})$.

▶ **Fact 16.** *There are effective meaning preserving translations from coalgebraic fixpoint formulas to modal automata and vice versa [17].*

4.3 Satisfiability and consequence game

In this subsection we introduce two of our main tools, viz., the satisfiability and consequence game associated with modal automata. Before we can turn to the definition of these games we need some preliminary notions.

▶ **Definition 17.** Fix a set A. We let $A^\sharp$ denote the set of binary relations over A, that is, $A^\sharp := \mathcal{P}(A \times A)$. A *trace* through a finite word $R_1R_2R_3...R_k$ over $A^\sharp$ is a finite A-word $a_0a_1a_2...a_k$ such that $a_iR_{i+1}a_{i+1}$ for all $i < k$. A trace through an $A^\sharp$-stream $R_1R_2R_3...$ is an A-stream $a_0a_1a_2...$ such that $a_iR_{i+1}a_{i+1}$ for all $i < \omega$. Given a parity map $\Omega : A \to \omega$, with $NBT(A, \Omega)$ we denote the set of $A^\sharp$-streams $R_1R_2R_3...$ that contains no *bad trace*, that is, no trace $a_0a_1a_2...$ such that the greatest parity occurring infinitely often is odd.

The satisfiability game $\mathcal{S}(\mathbb{A})$ for a modal automaton $\mathbb{A}$ as introduced in [3] is a two-player graph game played by $\exists$ and $\forall$. We want this game to be such that $\exists$ has a winning strategy in $\mathcal{S}(\mathbb{A})$ iff there is a pointed coalgebra $\mathbb{S}$ that is accepted by $\mathbb{A}$. The idea behind the satisfiability game is to make a simultaneous projection of all matches of the acceptance game on this pointed coalgebra. More in particular, every match of the satisfiability game can be seen as a *bundle* of matches of the acceptance game for the automaton $\mathbb{A}$ on $\mathbb{S}$. To

Table 3 Satisfiability Game.

Position	Player	Admissible moves
$R \in A^\sharp$	$\exists$	$\bigcap_{a \in \mathsf{ran}(R)} [\![\Theta(a)]\!]^1_a$
$(\mathtt{Y}, \alpha) \in \mathsf{T}_\mathtt{X}(A^\sharp)$	$\forall$	$\{R \mid R \subseteq R' \text{ for some } R' \in \mathit{Base}(\alpha)\}$

gather some intuition, suppose that a pointed coalgebra $\mathbb{S}$ is given and assume that $\exists$ has a winning strategy in the acceptance game for $\mathbb{A}$ and $\mathbb{S}$. Now we will see how $\exists$ can get a winning strategy in $\mathcal{S}(\mathbb{A})$.

Intuitively a basic position of $\mathcal{S}(\mathbb{A})$ should be represented as a subset B of A. We may associate with such a *macro-state* B a point $s \in S$ such that $\exists$ has to deal with positions (b, s) in the acceptance game, for all $b \in B$. For each $t \in S$ and for each $b \in B$, we define the set A^b_t as the collection of states $b' \in A$ such that (b', t) is a possible basic position in the acceptance game following the basic position (b, s). Since B is a macro-state, we define $A_t := \bigcup\{A^b_t \mid b \in B\}$. Hence, each such a set is a potential next combination of states in A that $\exists$ has to be able to handle simultaneously. In this set-up $\exists$'s move would be based on the set $\{A_t \mid t \in S\}$. Now it is up to $\forall$ to choose a set from this collection, moving to the next macro-state.

With this definition of the game, a match of $\mathcal{S}(\mathbb{A})$ corresponds to a sequence $\rho := B_0B_1B_2...$ of basic positions, which are subsets of A. Now to clarify whether ρ is won by $\exists$ we could naively say that $\exists$ wins if there is no bad trace $b_0b_1b_2...$ in ρ. However, if there is such a bad trace $b_0b_1b_2...$, this would only be a problem if it actually corresponds to a match of the acceptance game. Up to now we know that each b_i occurs in *some* match of the acceptance game, but there is no way to know whether $b_0b_1b_2...$ is the projection of an actual match of the acceptance game. This shows that defining the game based on subsets of A doesn't work properly. A solution to this problem is to replace the subset B by a relation $R \in A^\sharp$. The range of R would play the same role as B. This helps us to remember which traces are relevant, when we define the winning condition.

In the following we give the definition of satisfiability game and in Proposition 20 we state that as aimed for, $\exists$ has a winning strategy in $\mathcal{S}(\mathbb{A})$ iff there is a pointed coalgebra that is accepted by $\mathbb{A}$.

We first consider the one-step models based on the set $A^\sharp$ of binary relations over A.

▶ **Definition 18.** The *natural a-valuation* $V_a : A \to \mathcal{P}A^\sharp$ is given by

$$V_a : b \mapsto \{R \in A^\sharp \mid (a, b) \in R\}.$$

For $\alpha \in \mathsf{T}_\mathtt{X}A^\sharp$ and $\varphi \in \mathtt{1ML_T}(\mathtt{X}, A)$, we write $\alpha \Vdash^1_a \varphi$ to denote that $A^\sharp, \alpha \Vdash^1_{V_a} \varphi$, and we define $[\![\varphi]\!]^1_a := \{\alpha \in \mathsf{T}_\mathtt{X}A^\sharp \mid \alpha \Vdash^1_a \varphi\}$.

▶ **Definition 19.** The satisfiability game associated with a modal $\mathtt{X}$-automaton $\mathbb{A} = (A, \Theta, \Omega, a_I)$ is denoted by $\mathcal{S}(\mathbb{A})$ and given by Table 3.

Unless specified otherwise, we assume $\{(a_I, a_I)\}$ to be the starting position of $\mathcal{S}(\mathbb{A})$. An infinite match $R_1\alpha_1R_2\alpha_2R_3...$ is winning for $\exists$ if $R_1R_2R_3... \in \mathit{NBT}(A, \Omega)$.

▶ **Proposition 20** (Adequacy). *Let $\mathbb{A}$ be a modal automaton. Then $\exists$ has a winning strategy in $\mathcal{S}(\mathbb{A})$ iff the language recognized by $\mathbb{A}$ is non-empty [3].*

As announced in our abstract and introduction, an important role in our approach is played by the *consequence game* $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ associated with two automata $\mathbb{A}$ and $\mathbb{A}'$, which is

played by two players I and II. One may think of player II trying to show that automaton $\mathbb{A}$ *implies* $\mathbb{A}'$ by establishing a close structural connection between the two automata, and of player I trying to show this does not hold.

Matches of the consequence game $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ are tightly linked to the matches of the satisfiability games $\mathcal{S}(\mathbb{A})$ and $\mathcal{S}(\mathbb{A}')$ and this connection extends to the definition of the winning conditions of $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ in terms of winning conditions of $\mathcal{S}(\mathbb{A})$ and $\mathcal{S}(\mathbb{A}')$. In fact the consequence $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ is reminiscent of games defined by Santocanale, which go back to the literature on game semantics for linear logic (see [16] and references therein).

To describe the game we consider a match of $\mathcal{C}(\mathbb{A}, \mathbb{A}')$. Each round of this match consists of three moves. At the start of the round, at a basic position $(R, R') \in A^\sharp \times A'^\sharp$, player I picks a local model $\alpha \in \mathsf{T}_\mathtt{X} A^\sharp$ for formulas given by the range of R, as if she was player $\exists$ in the satisfiability game $\mathcal{S}(\mathbb{A})$. Second, player II transforms this one-step model into a model for formulas given by the range of R', inducing a move for $\exists$ in the satisfiability game $\mathcal{S}(\mathbb{A}')$. More precisely, player II provides a map $f : A^\sharp \to A'^\sharp$ turning α to a model for R'. The admissibility of this move reveals the essentially coalgebraic nature of the game, using the fact that T is actually a functor, i.e., it operates on arrows (that are, functions) as well as on objects (sets). More specfically, player II's move f is addmissible if the model α', that we obtain by applying the map $\mathsf{T}_\mathtt{X} f$ to the model α, is a model for R'. Player I then finishes the round by picking an element from the graph of map f as the next basic position.

▶ **Definition 21.** The *consequence game* $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ between modal automata $\mathbb{A} = (A, \Theta, \Omega, a_I)$ and $\mathbb{A}' = (A', \Theta', \Omega', a'_I)$ is given by the following table.

Table 4 Consequence Game.

Position	Player	Admissible moves
$(R, R') \in A^\sharp \times A'^\sharp$	I	$\bigcap_{a \in \mathsf{ran}(R)} [\![\Theta(a)]\!]^1_a$
$(\alpha, R') \in \mathsf{T}_\mathtt{X} A^\sharp \times A'^\sharp$	II	$\{f : A^\sharp \to A'^\sharp \mid \mathsf{T}_\mathtt{X} f(\alpha) \in \bigcap_{b \in \mathsf{ran}(R')} [\![\Theta'(b)]\!]^1_b\}$
$f : A^\sharp \to A'^\sharp$	I	$\{(R, R') \mid f(R) = R'\}$

As we already mentioned, pair of the form (R, R') in the above definition will be called a *basic position* of the consequence game. Similar to the satisfiability game, our standard assumption is that $(\{(a_I, a_I)\}, \{(a'_I, a'_I)\})$ is the starting position of $\mathcal{C}(\mathbb{A}, \mathbb{A}')$. We declare player I to be the winner of an infinite match $(R_1, R'_1)(R_2, R'_2)(R_2, R'_2)...$ if there exists a bad trace on the $\mathbb{A}'$-side, i.e. through $R'_1 R'_2 R'_3 ...$ but no bad trace on the $\mathbb{A}$-side i.e. through $R_1 R_2 R_3 ....$ In all other cases player II is the winner. Whenever II has a winning strategy in $\mathcal{C}(\mathbb{A}, \mathbb{A}')$ we say that $\mathbb{A}'$ is a *game consequence* of $\mathbb{A}$ and denote this fact with $\mathbb{A} \vDash_\mathcal{C} \mathbb{A}'$.

▶ **Proposition 22.** *Given automata* $\mathbb{A}$ *and* $\mathbb{A}'$ *we have that* $\mathbb{A} \vDash_\mathcal{C} \mathbb{A}'$ *implies* $L(\mathbb{A}) \subseteq L(\mathbb{A}')$.

Below Lemma 28 we shall see a counter-example to the converse of this proposition.

4.4 Formulas and automata

There are a few different methods for transforming a formula of the μ-calculus into an equivalent parity automaton [19][4]. The method used in [4] first constructs a tableau for the formula, which is then transformed into an automaton. Here, we shall instead use a direct translation from formulas to automata given by induction on the complexity of a formula [16]. We denote the translation of a formula φ by $\mathbb{A}_\varphi$. As a consequence of adequacy we get the following:

▶ **Fact 23.** *Given a formula φ and its translation $\mathbb{A}_\varphi$ we have that φ is satisfiable if and only if $\exists$ has a winning strategy in $\mathcal{S}(\mathbb{A})$ [3].*

For the opposite direction, we use a translation tr from automata to formulas. Unfortunately due to space limits we will not go through the definition of this translation, which is obtained using similar methods applied in [17]. The point about the translation tr is that it behaves well with respect to the notion of provability and enables us to apply proof-theoretic concepts, such as consistency, to automata. The key observation of this subsection is the following proposition.

▶ **Proposition 24.** *For every formula φ, we have $\varphi \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A}_\varphi)$.*

We will define the translation from formulas to automata by induction on the complexity of a formula. We omit the base step and inductive cases of disjunction, conjunction and negation, check [6] and [10] for details. For formulas of the form $\varphi = \nabla\alpha$ the automaton $\mathbb{A}_\varphi$ is defined by using formulas in $Base(\alpha)$ and applying T to the translation map.

For fixpoint formulas of the form $\varphi = \eta x.\alpha$ with $\eta \in \{\nu, \mu\}$, the construction of $\eta x.\mathbb{A}$ starts by putting $\mathbb{A}$ into a suitable shape denoted by $\mathbb{A}^x$. The key observation about $\mathbb{A}^x$ is that the free variable x in $\mathbb{A}$ becomes in a certain sense *guarded* in $\mathbb{A}^x$. Since we do not allow variables to appear guarded in the one-step formulas in the image of the transition map of an automaton, we need to introduce a new state $\underline{x}$ that we use to represent the variable x. For the construction of $\mathbb{A}^x$ we will use the fact that for every automaton $\mathbb{A}$ with a free variable x and any state $a \in A$, there are formulas θ_0^a and θ_1^a in which x does not appear, such that $\Theta(a) \equiv_{\mathbf{K}} (x \wedge \theta_0^a) \vee \theta_1^a$.
The following definition explains this auxiliary automaton:

▶ **Definition 25.** Let $\mathbb{A}$ be a modal automaton in which the variable x is free, and assume without loss of generality, that the priority map Ω is injective, and the smallest priority in the image of Ω is greater than 0. Pick a new state $\underline{x} \notin A$. Then we define the automaton $\mathbb{A}^x = (A^x, \Theta^x, a_I^x, \Omega^x)$ as follows:

- $A^x = (A \times \{0,1\}) \cup \{\underline{x}\}$. We write (a,i) as a_i, for $i \in \{0,1\}$.
- $\Theta^x(a_0) = \theta_0^a[\kappa]$, $\Theta^x(a_1) = \theta_1^a[\kappa]$ and $\Theta^x(\underline{x}) = x$,
- $\Omega^x(a_i) = \Omega(a)$, $\Omega^x(\underline{x}) = 0$ and $a_I^x = (a_I)_1$.

Here, κ is defined to be the substitution $a \mapsto (\underline{x} \wedge a_0) \vee a_1$ for every a.

We are now ready to define fixpoint operations on automata:

▶ **Definition 26.** The automaton $\mu x.\mathbb{A} = (A', \Theta', \Omega', a_I')$ is defined by setting $A' = A^x$, $\Theta'(a_i) = \Theta^x(a_i)$ for $a \in A$, $\Theta'(\underline{x}) = \theta_1^{a_I}[\kappa]$, $a_I' = \underline{x}$, $\Omega'(a_i) = \Omega^x(a_i)$ and $\Omega^x(\underline{x}) = 2 \cdot \mathtt{max}(\Omega^x[A^x]) + 1$. The automaton $\nu x.\mathbb{A} = (A', \Theta', \Omega', a_I')$ is defined in the same way, except that $\Theta'(\underline{x}) = \theta_0^{a_I}[\kappa] \vee \theta_1^{a_I}[\kappa]$ and $\Omega^x(\underline{x}) = 2 \cdot \mathtt{max}(\Omega^x[A^x]) + 2$.

In the following we define the notion of substitution for modal automata.

▶ **Definition 27.** Let $\mathbb{A} = (A, \Theta_A, \Omega_A, a_I)$ and $\mathbb{B} = (B, \Theta_B, \Omega_B, b_I)$ be modal automata over the languages P and $P \setminus \{x\}$, respectively. Assume that $\mathbb{A}$ is positive in x. We define the modal $P \setminus \{x\}$-automaton $\mathbb{A}[\mathbb{B}/x]$ as the structure $(D, \Theta_D, \Omega_D, d_I)$, where $D := A \uplus B$, $d_I := a_I$, and the transition map Θ_D is given by

$$\Theta_D(d) := \begin{cases} \Theta_A(d)[\Theta_B(b_I)/x] & \text{if } d \in A \\ \Theta_B(d) & \text{if } d \in B, \end{cases}$$

and $\Omega_D := \Omega_A \uplus \Omega_B$.

The next lemma lists some of the properties of the automaton $\mu x.\mathbb{A}$:

▶ **Lemma 28.** *Given modal automata $\mathbb{A}$ with x free, we have*
1. $\mathsf{tr}(\mu x.\mathbb{A}) \equiv_{\mathbf{K}} \mu x.\mathsf{tr}(\mathbb{A}^x)$;
2. $\mathbb{A}^x[\mu x.\mathbb{A}/x] \vDash_{\mathcal{C}} \mu x.\mathbb{A}$.

In passing we note that if in the second item of the above lemma we replace $\mathbb{A}^x$ with $\mathbb{A}$, then it is generally not the case that $\mathbb{A}[\mu x.\mathbb{A}/x] \vDash_{\mathcal{C}} \mu x.\mathbb{A}$. This is a counter-example to the converse of Proposition 22.

The following lemma summarizes properties of the translation tr which are needed to prove Proposition 24.

▶ **Lemma 29.** *There exists a translation tr from modal automata to the formulas of $\mu\mathtt{ML}_{\mathsf{T}}$ and operators $\wedge, \vee, \neg$ and ∇ on automata such that the following claims hold:*
1. *For all $\mathbb{A}, \mathbb{B}$, $\mathsf{tr}(\mathbb{A} \wedge \mathbb{B}) \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A}) \wedge \mathsf{tr}(\mathbb{B})$, $\mathsf{tr}(\mathbb{A} \vee \mathbb{B}) \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A}) \vee \mathsf{tr}(\mathbb{B})$ and $\mathsf{tr}(\neg\mathbb{A}) \equiv_{\mathbf{K}} \neg\mathsf{tr}(\mathbb{A})$;*
2. *Given automata $\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_n$ and $\mathfrak{a} \in \mathsf{T}\{\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_n\}$ then $\mathsf{tr}(\nabla\mathfrak{a}) \equiv_{\mathbf{K}} \nabla(\mathsf{T}\mathsf{tr})(\mathfrak{a})$;*
3. *For every automaton $\mathbb{A}$ with x free in $\mathbb{A}$ and $\eta \in \{\nu, \mu\}$ we have $\mathsf{tr}(\eta x.\mathbb{A}) \equiv_{\mathbf{K}} \eta x.\mathsf{tr}(\mathbb{A})$;*
4. *For all $\mathbb{A}, \mathbb{B}$ with x free in $\mathbb{A}$, we have $\mathsf{tr}(\mathbb{A}[\mathbb{B}/x]) \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A})[\mathsf{tr}(\mathbb{B})/x]$.*

5 Disjunctive and semi-disjunctive automata

Generally, the combinatorics of the trace graph(s) associated with the satisfiability and the consequence games are rather involved. As mentioned in the introduction, an important role in our proof of the Kozen-Walukiewicz theorem is played by two kinds of special automata that allow somewhat simpler trace graphs: *disjunctive* and *semi-disjunctive* automata. The conditions on these automata can be nicely expressed in terms of restrictions on the one-step language.

▶ **Definition 30.** Let $\mathbb{A}$ be a modal automaton and let C be a cluster of $\mathbb{A}$. An element $a \in C$ is called a *maximal even element* of C if it has the maximal priority in C, and this priority is even. A relation $R \in A^\sharp$ is *thin with respect to* $\mathbb{A}$ and a if:
1. for all $b \in A$ with aRb we have $a \lhd_{\mathbb{A}} b$;
2. for all $b_1, b_2 \in A$ with $b_1, b_2 \in R[a] \cap C_a$, either $b_1 = b_2$ or one of b_1 and b_2 is a maximal even element of C_a.

We call R $\mathbb{A}$-thin or simply thin, if it is thin with respect $\mathbb{A}$ and all $a \in A$.

A motivating observation about thin relations is the following.

▶ **Fact 31.** *For a stream $\rho = R_1R_2R_3...$ of thin relations there exists a finite collection F of traces on ρ such that any trace t on ρ is bad if and only if there is some $t' \in F$ cofinally equal to t.*

We are now ready to define *disjunctive* and *semi-disjunctive* automata.

▶ **Definition 32.** Given sets $\mathtt{X}$ and A we define the sets $\mathtt{LitC}(X)$ and $\mathtt{1ML}^d_{\mathsf{T}}(\mathtt{X}, A)$ by respectively:

$$\pi ::= \bot \mid \top \mid p \wedge \pi \mid \neg p \wedge \pi$$

and

$$\alpha ::= \bot \mid \pi \wedge \nabla\beta \mid \alpha \vee \alpha,$$

where $\pi \in \mathtt{LitC}(\mathtt{X})$ and $\beta \in \mathsf{T}A$. Elements of $\mathtt{1ML}^d_{\mathsf{T}}(\mathtt{X}, A)$ are called *one-step disjunctive formulas* and a modal automaton $\mathbb{A} = (A, \Theta, \Omega, a_I)$ is *disjunctive* if $\Theta(a)$ belongs to $\mathtt{1ML}^d_{\mathsf{T}}(\mathtt{X}, A)$ for all $a \in A$.

▶ **Definition 33.** Let $\mathbb{A}$ be a modal automaton an let C be a cluster of $\mathbb{A}$. The set of *(zero-step) C-safe conjunctions*, denoted by $\mathtt{Conj}_0^C(A)$ contains formulas of the form $\bigwedge B$ with $B \subseteq A$, such that for all $b_1 \neq b_2 \in B \cap C$, either b_1 or b_2 is a maximal even element of C. The grammar

$$\alpha ::= \bot \mid \pi \wedge \nabla\gamma \mid \alpha \vee \alpha,$$

where $\pi \in \mathtt{LitC(X)}$ and $\gamma \in \mathtt{TConj}_0^C(A)$, defines the set $\mathtt{1ML}_{\mathsf{T}}^{s(C)}(\mathtt{X}, A)$ of *one-step C-safe formulas*. We call a one-step formula α *semi-disjunctive with respect to $a \in A$* if α is a C_a-safe formula. A modal automaton $\mathbb{A} = (A, \Theta, \Omega, a_I)$ is *semi-disjunctive* if $\Theta(a)$ is semi-disjunctive with respect to a for all $a \in A$.

The key property of these automata is that the matches of the satisfiability and consequence game are of a relatively simple shape. For disjunctive (respectively semi-disjunctive) automata, without loss of generality, we can always assume that these matches contain only functional (respectively thin) relations. This property allows us to work with a variant of the satisfiability game which is called *thin* satisfiability game.

▶ **Definition 34.** The *thin* satisfiability game $\mathcal{S}_{thin}(\mathbb{A})$ is the variant of the satisfiability game $\mathcal{S}(\mathbb{A})$ where $\forall$'s choice of moves is restricted to *thin* relations. A winning strategy for $\forall$ in $\mathcal{S}_{thin}(\mathbb{A})$ is called a *thin refutation* of $\mathbb{A}$.

Note that for an arbitrary automaton $\mathbb{A}$ it is not always the case that $\mathcal{S}(\mathbb{A})$ and $\mathcal{S}_{thin}(\mathbb{A})$ are equivalent, but for disjunctive and semi-disjunctive automata it holds:

▶ **Proposition 35.** *Given a semi-disjunctive automaton $\mathbb{A}$, then $\exists$ ($\forall$, respectively) has a winning strategy in $\mathcal{S}(\mathbb{A})$ if and only if $\exists$ ($\forall$, respectively) has a winning strategy in $\mathcal{S}_{thin}(\mathbb{A})$.*

In our paper we will use the construction given by [10, Definition 3.10] to transform a modal automaton $\mathbb{A}$ into a disjunctive automaton denoted by $\mathsf{sim}(\mathbb{A})$[3]. The proof of equivalence of $\mathbb{A}$ and $\mathsf{sim}(\mathbb{A})$ amounts to a coalgebraic version of the simulation theorem [13] for modal automata [10]. Roughly speaking, the idea of the proof is to define $\mathsf{sim}(\mathbb{A})$ via a variation of the power set construction such that a match of the acceptance game of $\mathsf{sim}(\mathbb{A})$ corresponds to $\exists$ simultaneously playing various matches of the acceptance game $\mathbb{A}$.

▶ **Fact 36.** *Let $\mathbb{A}$ be a modal automaton. Then $\mathsf{sim}(\mathbb{A})$ is a disjunctive automaton satisfying $L(\mathbb{A}) = L(\mathsf{sim}(\mathbb{A}))$.*

▶ **Proposition 37.** *The map $\mathsf{sim}(\cdot)$ assigns to each modal automaton $\mathbb{A}$ a disjunctive automaton $\mathsf{sim}(\mathbb{A})$ such that:*

1. *$\mathbb{A}$ and $\mathsf{sim}(\mathbb{A})$ are semantically equivalent;*
2. *$\mathbb{A} \vDash_{\mathcal{C}} \mathsf{sim}(\mathbb{A})$.*

The rest of this section is devoted to discuss other properties of disjunctive and semi-disjunctive automata which will later on be used to prove our main technical result, viz., Theorem 1. In the following theorem we show how to use the fact that an automaton $\mathbb{D}$ is a game consequence of an automaton $\mathbb{A}$, to find a thin refutation in the satisfiability game for $\mathbb{A} \wedge \neg\mathbb{D}$.

[3] Note that the approach in [10] does not explicitly use the ∇ operator but the automata in [10] can be seen as a notational variant of the ones employed here.

▶ **Theorem 38.** *Let $\mathbb{A}$ be a semi-disjunctive automaton and $\mathbb{D}$ be an arbitrary modal automaton such that $\mathbb{A} \vDash_{\mathcal{C}} \mathbb{D}$. Then $\mathbb{A} \wedge \neg\mathbb{D}$ has a thin refutation.*

This theorem is an automata-theoretic version of Lemma 36 from [18], one of the key lemmas of Walukiewicz' completeness proof, and at the same time it generalizes that result in two ways; first, our coalgebraic approach extends the result from the power set functor $\mathcal{P}$ to a set functor T, second, we prove the result for an arbitrary automaton $\mathbb{D}$ instead for a disjunctive one.

Proof sketch of Theorem 38. To fix notation, let $\mathbb{A} = (A, \Theta, \Omega, a_I)$ and $\mathbb{D} = (D, \Theta', \Omega', d_I)$. We recall that the transition map of the automaton $\neg\mathbb{D}$ is defined by taking boolean duals of the formulas assigned by the transition map of $\mathbb{D}$, while the priority map is defined by simply raising all priorities by 1.

Assume that $\mathbb{A} \vDash_{\mathcal{C}} \mathbb{D}$. Without loss of generality we can assume that II has a winning strategy involving thin relations only. To show that the automaton $\mathbb{A} \wedge \neg\mathbb{D}$ has a thin refutation, we will define a thin winning strategy χ for $\forall$ in $\mathcal{S}_{thin}(\mathbb{A} \wedge \neg\mathbb{D})$. Given a χ-guided partial match in $\mathcal{S}(\mathbb{A} \wedge \neg\mathbb{D})$ with basic positions $R_0R_1R_2...R_n$. Our aim is to introduce a response R_{n+1} for $\forall$ to every possible move γ by $\exists$, such that:

(i) R_{n+1} is a legitimate move, i.e., $R_{n+1} \subseteq R'$ for some $R' \in \mathit{Base}(\gamma)$;
(ii) $\mathsf{ran}(R_{n+1}) \cap D$ is a singleton;
(iii) R_{n+1} is thin.

We shall also maintain the induction hypothesis that for every χ-guided partial match $R_0, ..., R_n$ there is a shadow-match in the consequence game, guided by the winning strategy for player II, of the form $(S_0, S_0')(S_1, S_1')...(S_n, S_n')$ where, for each i, we have $R_i \cap A^2 = S_i$ and $R_i \cap D^2 \subseteq S_i'$.

Going to the details of how we maintain these condition, we claim that:

▶ **Claim.** *There is some $S \in \mathit{Base}(\gamma)$ and some $c \in D$ with $(d, c) \in f(S \cap A^2) \cap (S \cap D^2)$, where $f : A^\sharp \to D^\sharp$ is dictated by Player II's winning strategy in $\mathcal{C}(\mathbb{A}, \mathbb{D})$.*

Proof is an exercise in one-step coalgebraic logic.

With this claim in place, we define the next move for $\forall$ by picking the relation $(S \cap A^2) \cup \{(d, c)\}$, where $S \in \mathit{Base}(\gamma)$ and $c \in D$ are as described in the claim, so that $(d, c) \in f(S \cap A^2) \cap (S \cap D^2)$. Then we prove that this a legitimate move for $\forall$ and the shadow-match which is then extended by the pair $(S \cap A^2, f(S \cap A^2))$ also satisfies condition (i)-(iii). Thin-ness of the strategy χ defined in this way follows from semi-disjunctivity of the automaton $\mathbb{A}$. Finally we show that every infinite χ-guided match ρ contains a bad trace. Note that any infinite match contains a unique trace in D, which will also be a trace on the right side of the shadow-match in the consequence game. If this trace is *not* bad given the priorities assigned to states in $\neg\mathbb{D}$, then it must be bad as a trace in $\mathbb{D}$ since parities are swapped in $\neg\mathbb{D}$. So there must be a bad trace on the left side of the shadow-match in the consequence game, since this shadow-match was guided by the winning strategy of II. But since every such trace corresponds to a trace through A in the satisfiability game, we see that either the unique trace through D in ρ is bad or there is some bad trace through A in ρ. In either case, there must be some bad trace in ρ, so $\forall$ wins. ◀

The next Theorem is a version of another key lemma in Walukiewicz' proof for modal automata, viz., [18, Lemma 39].

▶ **Theorem 39.** *Let $\mathbb{A}$ and $\mathbb{B}$ be arbitrary modal automata, and let x be a free propositional variable of $\mathbb{B}$. Then we have $\mathbb{B}[\mathsf{sim}(\mathbb{A})/x] \vDash_{\mathcal{C}} \mathbb{B}[\mathbb{A}/x]$.*

Proof sketch of Theorem 39. Starting with notation, let $\mathbb{A} = (A, \Theta_A, \Omega_A, a_I)$, $\mathbb{B} = (B, \Theta_B, \Omega_B, b_I)$ and $\mathsf{sim}(\mathbb{A}) = \mathbb{D} = (D, \Theta_D, \Omega_D, d_I)$. From the proof of Proposition 37 we get a map $G : D \to A^\sharp$ that *reflects traces* in the sense that if $G(d_i)_{i\in\omega} \in (A^\sharp)^\omega$ contains a bad $\mathbb{A}$-trace, then $(d_i)_{i\in\omega}$ is itself a bad $\mathbb{D}$-trace. By the proof of Proposition 37 part (2) this map G encodes a particularly simple winning strategy for player II in the consequence game $\mathcal{C}(\mathbb{D}, \mathbb{A})$. The trick of proving Theorem 39 is to turn this winning strategy encoded by G into a new winning strategy for player II in $\mathcal{C}(\mathbb{B}[\mathbb{D}/x], \mathbb{B}[\mathbb{A}/x])$.

Let us first clarify why the proof is not so straightforward. To see where the difficulties lie, consider an arbitrary infinite match $\rho = (R_n, R'_n)_{n\in\omega}$ of the consequence game for $\mathbb{B}[\mathbb{D}/p]$ and $\mathbb{B}[\mathbb{A}/p]$. Given the shape of these two automata, we may assume that traces on $\rho_l := R_0R_1\ldots$ consist of either a $\mathbb{B}$-stream, or of a finite $\mathbb{B}$-trace followed by an infinite $\mathbb{D}$-trace, and that traces on $\rho_r := R'_0R'_1\ldots$ are either a $\mathbb{B}$-stream or composed of a finite $\mathbb{B}$-trace and an $\mathbb{A}$-trace. Our purpose will be to associate with each ρ_r-trace $\tau = b_0b_1\ldots b_na_{n+1}a_{n+2}a_{n+3}\ldots$, a ρ_l-trace $\tau_l = b_0b_1\ldots b_nd_{n+1}d_{n+2}d_{n+3}\ldots$, such that we can use the trace reflection on the $\mathbb{D}$- and $\mathbb{A}$-tail of τ and τ_l, respectively.

For this purpose we will define, for each partial match leading to final position (R_n, R'_n), a map $g_n : \mathsf{ran}_A R'_n \to \mathsf{ran}_D R_n$. Intuitively, for $a \in A$, $g_n(a)$ represents a $d \in D$ that 'implies' a, more precisely $a \in \mathsf{ran}G(g_na)$. Ideally, we would like to show that the τ-tail $(a_i)_{i>n}$ is in fact a trace on the $A^\sharp$-stream $(G(g_ia_i))_{i>n}$, while $(g_ia_i)_{i>n}$ is an ρ_l-trace so that the trace reflection applies indeed.

Unfortunately, this is too good to be true, due to complications that are caused by $\mathbb{A}$-traces *merging*: the point is that *trace jumps* may occur, that is, situations where for some pair $(a, a') \in R'_{j+1}$ it does not hold that $(g_ja, g_{j+1}a') \in R_{j+1}$. Our solution to this problem will be to ensure that every ρ_r-trace can suffer only *finitely many* trace jumps. Thus, what we *can* show is that any $\mathbb{A}$-trace $a_0a_1\cdots$ has a *tail* $a_ka_{k+1}a_{k+2}\cdots$ which is a trace on $G(g_ka_k)G(g_{k+1}a_{k+1})G(g_{k+2}a_{k+2})\cdots$ This suffices to prove that if there is a bad trace on ρ_r, then there is also a bad trace on ρ_l, so that player II indeed wins the match ρ.

The *tool* that we employ to guarantee this consists of a well-founded ordering of the collection of those ρ_l-traces that arrive to the $\mathbb{D}$-part of the automaton $\mathbb{B}[\mathbb{D}/p]$. The definition of this ordering crucially uses the disjunctivity of $\mathbb{D}$.

We define strategy χ by a simultaneous induction on the length of a partial χ-match $\rho = (R_0, R'_0), \ldots, (R_n, R'_n)$. Using a well-founded order on the set of traces, and as we already mentioned, we define a map $g_n : \mathsf{ran}_A R'_n \to \mathsf{ran}_D R_n$ and a map $F_n : (B \cup D)^\sharp \to (B \cup A)^\sharp$. We let the F-maps determine player II's strategy in the following sense. Suppose that in the mentioned partial match ρ, player I legitimately picks an element $(\Upsilon, \alpha) \in \mathsf{T}(B \cup D)^\sharp$. Then player II's response will be the map $F_n \restriction_{Base(\alpha)}$.

Inductively we maintain the following conditions.

(∗) $F_{n-1}R_n = R'_n$, for all $n > 0$;
(†1) $R'_n \cap (B \times B) \subseteq R_n$;
(†2) $R'_n \cap (B \times A) \subseteq \bigcup_{d\in D}\{(b, a) \mid (b, d) \in R_n \text{ and } a \in \mathsf{ran}G(d)\}$;
(†3) $R'_n \cap (A \times A) \subseteq \bigcup\{G(d) \mid d \in \mathsf{ran}R_n \cap D\}$;
(††) $a \in \mathsf{ran}G(g_n(a))$ for all $a \in \mathsf{ran}R'_n \cap A$.

These conditions enable us to keep track of the shape of $\mathbb{B}[\mathbb{D}/p]$- and $\mathbb{B}[\mathbb{A}/p]$-traces.

Applying the tool that we developed for *trace management* we can extend the match and define g_{n+1} and thus F_{n+1} in such a way that conditions (∗)–(††) remain true for one more round. We finish the proof by showing that the following claims hold:

▶ **Claim** (1). *The moves for player II prescribed by the strategy χ are legitimate.*

Proof of this claim is straightforward according to properties of one-step coalgebraic logic.

▶ **Claim** (2). *Suppose ρ is an infinite χ-guided match with basic positions*

$$(R_0, R'_0)(R_1, R'_1)(R_2, R'_2)\dots.$$

If there is a bad trace on $R'_0R'_1R'_2...$, there is also a bad trace on $R_0R_1R_2...$.

Proof is based on the actual definition of maps g_n and trace reflection property of map G. ◀

The final lemma of this section summarizes some of the closure properties of (semi)-disjunctive automata.

▶ **Lemma 40.** *Let $\mathbb{A}$ and $\mathbb{B}$ be modal automata. Then we have:*
1. *if $\mathbb{A}$ is disjunctive, then it is also semi-disjunctive;*
2. *if $\mathbb{A}$ and $\mathbb{B}$ are disjunctive, then so is $\mathbb{A} \vee \mathbb{B}$;*
3. *if $\mathbb{A}$ and $\mathbb{B}$ are semi-disjunctive, then so are $\mathbb{A} \vee \mathbb{B}$ and $\mathbb{A} \wedge \mathbb{B}$;*
4. *if $\mathbb{A}$ is semi-disjunctive and $\mathbb{B}$ is disjunctive, then $\mathbb{A}[\mathbb{B}/x]$ is semi-disjunctive;*
5. *$\nu x.\mathbb{A}$ and $\mathbb{A}^x$ are semi-disjunctive in case $\mathbb{A}$ is disjunctive.*

For the clauses (3)-(5) of this lemma we need to involve the modal distribution laws in order to make sure that all constructed automata have the right syntactic shape.

6 Completeness

In this section we give an overview of the completeness proof for $\mu\mathtt{ML}_\mathsf{T}$ with respect to the derivation system **K**. In [7] Kozen proved the completeness of his proof system for a fragment of the modal μ-calculus: he showed that for *aconjunctive* formulas consistency implies satisfiability. The following lemma can be seen as an automata-theoretic version of Walukiewicz' rendering of this result.

▶ **Lemma 41.** *Given an automaton $\mathbb{A}$, if $\mathsf{tr}(\mathbb{A})$ is consistent, then $\exists$ has a winning strategy in the thin satisfiability game for $\mathbb{A}$.*

We now turn to the proof of our main technical result, viz., Theorem 1, as we discussed in the introduction.

To prove this theorem, recall from Proposition 24 that every formula $\varphi \in \mu\mathtt{ML}_\mathsf{T}$ is provably equivalent to the translation of the modal automaton $\mathbb{A}_\varphi$. Thus in particular we have $\vdash_\mathbf{K} \varphi \to \mathsf{tr}(\mathbb{A}_\varphi)$. We now want to apply the automata-theoretic machinery that we developed in previous sections, to strengthen this result, showing that for any formula φ there is a *disjunctive* automaton $\mathbb{D}_\varphi$ such that $\vdash_\mathbf{K} \varphi \to \mathsf{tr}(\mathbb{D}_\varphi)$. The following lemma shows that whenever φ is the translation of a semi-disjunctive automaton this result can be proved.

▶ **Lemma 42.** *Let $\mathbb{A}$ be any semi-disjunctive automaton. Then $\vdash_\mathbf{K} \mathsf{tr}(\mathbb{A}) \to \mathsf{tr}(\mathsf{sim}(\mathbb{A}))$.*

Proof. By Proposition 37 there is a winning strategy for player II in the consequence game $\mathcal{C}(\mathbb{A}, \mathsf{sim}(\mathbb{A}))$. Since $\mathbb{A}$ is semi-disjunctive, it follows from Proposition 38 that there is a winning strategy for $\forall$ in the thin satisfiability game for $\mathbb{A} \wedge \neg\mathsf{sim}(\mathbb{A})$. We get $\vdash_\mathbf{K} \neg\mathsf{tr}(\mathbb{A} \wedge \neg\mathsf{sim}(\mathbb{A}))$ by Lemma 41. Then from clauses (1) of Lemma 29 we have $\vdash_\mathbf{K} \neg(\mathsf{tr}(\mathbb{A}) \wedge \neg\mathsf{tr}(\mathsf{sim}(\mathbb{A})))$, which means that $\vdash_\mathbf{K} \mathsf{tr}(\mathbb{A}) \to \mathsf{tr}(\mathsf{sim}(\mathbb{A}))$ as required. ◀

Proof of Theorem 1. The proof is given by induction on the complexity of formula φ. We assume without loss of generality that φ is in negation normal form and inductively omit the cases of literals, disjunctions and the modal operator. For conjunctions, given formulas α, α' we have disjunctive automata $\mathbb{D} \equiv \alpha$ and $\mathbb{D}' \equiv \alpha'$ such that $\vdash_{\mathbf{K}} \alpha \to \mathsf{tr}(\mathbb{D})$ and $\vdash_{\mathbf{K}} \alpha' \to \mathsf{tr}(\mathbb{D}')$. By the first clause of Lemma 29 we get $\vdash_{\mathbf{K}} \alpha \wedge \alpha' \to \mathsf{tr}(\mathbb{D} \wedge \mathbb{D}')$. But $\mathbb{D} \wedge \mathbb{D}'$ is semi-disjunctive by the third clause of Lemma 40, and we can apply Lemma 42 to obtain the desired conclusion.

Finally we turn to the fixpoint operators. Given a formula $\varphi = \nu x.\alpha(x)$ and let x be a free variable of α. Inductively there is a disjunctive automaton $\mathbb{A}$ for α such that $\vdash_{\mathbf{K}} \alpha(x) \to \mathsf{tr}(\mathbb{A})$. Since $\mathbb{A}$ is disjunctive, by the last clause of Lemma 40 $\nu x.\mathbb{A}$ is semi-disjunctive, so it suffices to show that $\vdash_{\mathbf{K}} \nu x.\alpha(x) \to \mathsf{tr}(\nu x.\mathbb{A})$. By clause (3) of Lemma 29, it suffices to prove that $\vdash_{\mathbf{K}} \nu x.\alpha(x) \to \nu x.\mathsf{tr}(\mathbb{A})$, and this clearly follows from our assumption that $\vdash_{\mathbf{K}} \alpha(x) \to \mathsf{tr}(\mathbb{A})$.

Now we consider the crucial case where $\varphi = \mu x.\alpha(x)$. By the induction hypothesis there is a semantically equivalent disjunctive automaton $\mathbb{A}$ for $\alpha(x)$ such that $\vdash_{\mathbf{K}} \alpha(x) \to \mathsf{tr}(\mathbb{A})$. Let $\mathbb{D} := \mathsf{sim}(\mu x.\mathbb{A})$. This automaton is clearly semantically equivalent to φ. We want to show that:

$$\vdash_{\mathbf{K}} \mu x.\mathsf{tr}(\mathbb{A}) \to \mathsf{tr}(\mathbb{D}),$$

from which the result follows since $\vdash_{\mathbf{K}} \varphi \to \mu x.\mathsf{tr}(\mathbb{A})$. By clause (1) of Lemma 28 together with clause (3) of Lemma 29 we get $\mu x.\mathsf{tr}(\mathbb{A}) \equiv_{\mathbf{K}} \mu x.\mathsf{tr}(\mathbb{A}^x)$, so it in fact suffices to prove:

$$\vdash_{\mathbf{K}} \mu x.\mathsf{tr}(\mathbb{A}^x) \to \mathsf{tr}(\mathbb{D}).$$

Hence by the fixpoint rule it suffices to prove that:

$$\vdash_{\mathbf{K}} \mathsf{tr}(\mathbb{A}^x)[\mathsf{tr}(\mathbb{D})/x] \to \mathsf{tr}(\mathbb{D}).$$

But using clause (4) of Lemma 29 we get

$$\mathsf{tr}(\mathbb{A}^x)[\mathsf{tr}(\mathbb{D})/x] \equiv_{\mathbf{K}} \mathsf{tr}(\mathbb{A}^x[\mathbb{D}/x]),$$

so it suffices to prove $\vdash_{\mathbf{K}} \mathsf{tr}(\mathbb{A}^x[\mathbb{D}/x]) \to \mathsf{tr}(\mathbb{D})$, or equivalently:

$$\vdash_{\mathbf{K}} \neg(\mathsf{tr}(\mathbb{A}^x[\mathbb{D}/x]) \wedge \neg\mathsf{tr}(\mathbb{D})).$$

We can now apply the clauses (2) and (3) of Lemma 29 to see that this is equivalent to $\vdash_{\mathbf{K}} \neg\mathsf{tr}(\mathbb{A}^x[\mathbb{D}/x] \wedge \neg\mathbb{D})$, and by Lemma 41 it therefore suffices to prove that $\forall$ has a winning strategy in the thin satisfiability game for the automaton $\mathbb{A}^x[\mathbb{D}/x] \wedge \neg\mathbb{D}$. Note that by clause (5) of Lemma 40 $\mathbb{A}^x$ is semi-disjunctive since $\mathbb{A}$ is disjunctive. Now since $\mathbb{D}$ is disjunctive and $\mathbb{A}^x$ is semi-disjunctive, from Lemma 40 clause (4) it follows that $\mathbb{A}^x[\mathbb{D}/x]$ is semi-disjunctive too. Hence, by Theorem 38 the required conclusion follows if we can show that $\mathbb{A}^x[\mathbb{D}/x] \models_{\mathcal{C}} \mathbb{D}$. But from Lemma 28 and Proposition 37 we get by transitivity of game consequance:

$$\mathbb{A}^x[\mu x.\mathbb{A}/x] \models_{\mathcal{C}} \mu x.\mathbb{A} \models_{\mathcal{C}} \mathsf{sim}(\mu x.\mathbb{A}) = \mathbb{D}.$$

so it suffices to show that

$$\mathbb{A}^x[\mathbb{D}/x] \models_{\mathcal{C}} \mathbb{A}^x[\mu x.\mathbb{A}/x].$$

But this is an instance of Theorem 39, and so we are done. ◀

Finally we see how Theorem 1 implies completeness.

▶ **Theorem 43** (Completeness). *Every consistent formula $\varphi \in \mu\mathtt{ML}_{\mathsf{T}}$ is satisfiable.*

Proof. Given a consistent formula φ, by Theorem 1 there exists a semantically equivalent disjunctive automaton $\mathbb{D}$ such that $\mathsf{tr}(\mathbb{D})$ is consistent too. Now by Lemma 41 $\exists$ has a winning strategy in $\mathcal{S}_{thin}(\mathbb{D})$. But $\mathbb{D}$ is disjunctive and hence semi-disjunctive, and so by Proposition 35 $\exists$ also has a winning strategy in $\mathcal{S}(\mathbb{D})$. ◀

Acknowledgement. The authors would like to thank the anonymous referees for many useful comments.

References

1 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

2 Giovanna D'Agostino and Marco Hollenberg. Logical questions concerning the μ-calculus: Interpolation, lyndon and łoś-tarski. *Journal of Symbolic Logic*, 65(1):310–332, 2000.

3 Gäelle Fontaine, Raul Leal, and Yde Venema. Automata for coalgebras: An approach using predicate liftings. In *Automata, Languages and Programming: 37th International Colloquium ICALP'10*, pages 381–392, 2010.

4 David Janin and Igor Walukiewicz. Automata for the modal μ-calculus and related results. In *20th Symposium on Mathematical Foundations of Computer Science MFCS'95*, volume 969, pages 552–562, 1995.

5 David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *7th International Conference on Concurrency Theory CONCUR'96*, pages 263–277, 1996.

6 Christian Kissig and Yde Venema. Complementation of coalgebra automata. In *3rd Conference on Algebra and Coalgebra in Computer Science CALCO'09*, pages 81–96, 2009.

7 Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

8 Clemens Kupke, Alexander Kurz, and Yde Venema. Completeness of the finitary Moss logic. In *9th Conference on Advances in Modal Logic AiML'08*, pages 193–217, 2008.

9 Clemens Kupke, Alexander Kurz, and Yde Venema. Completeness for the coalgebraic cover modality. *Logical Methods in Computer Science*, 8:1–76, 2012.

10 Clemens Kupke and Yde Venema. Coalgebraic automata theory: Basic results. *Logical Methods in Computer Science*, 4(4), 2008.

11 Johannes Marti, Fatemeh Seifan, and Yde Venema. Uniform interpolation for coalgebraic fixpoint logic. In *6th Conference on Algebra and Coalgebra in Computer Science CALCO'15*, pages 238–252, 2015.

12 Larry Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999. (Erratum published *Ann.P.Appl.Log.* 99:241–259, 1999).

13 David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.

14 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

15 J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

16 Luigi Santocanale. Free μ-lattices. *Journal of Pure and Applied Algebra*, 168(2-3):227–264, 2002. Category Theory 1999: selected papers, conference held in Coimbra in honour of the 90th birthday of Saunders Mac Lane.

17 Yde Venema. Automata and fixed point logic: a coalgebraic perspective. *Information and Computation*, 204:637–678, 2006.

18 Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157:142–182, 2000.

19 Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bulletin of the Belgian Mathematical Society – Simon Stevin*, 8(2):359–391, 2001.

AC Dependency Pairs Revisited*

Akihisa Yamada[1], Christian Sternagel[2], René Thiemann[3], and Keiichirou Kusakari[4]

1 **University of Innsbruck, Austria**
2 **University of Innsbruck, Austria**
3 **University of Innsbruck, Austria**
4 **Gifu University, Japan**

Abstract

Rewriting modulo AC, i.e., associativity and/or commutativity of certain symbols, is among the most frequently used extensions of term rewriting by equational theories. In this paper we present a generalization of the dependency pair framework for termination analysis to rewriting modulo AC. It subsumes existing variants of AC dependency pairs, admits standard dependency graph analyses, and in particular enjoys the minimality property in the standard sense. As a direct benefit, important termination techniques are easily extended; we describe usable rules and the subterm criterion for AC termination, which properly generalize the non-AC versions.

We also perform these extensions within IsaFoR – the Isabelle formalization of rewriting – and thereby provide the first formalization of AC dependency pairs. Consequently, our certifier CeTA now supports checking proofs of AC termination.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs, F.4.1 Mathematical Logic, F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases Equational Rewriting, Termination, Dependency Pairs, Certification

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.8

1 Introduction

The *dependency pair (DP) method* of Arts and Giesl [2] and its successor, the *DP framework* [7], have become the de facto standard in termination proving for term rewrite systems, providing the foundation of state-of-the-art termination provers.

Various authors extended the DP method/framework for proving termination *modulo AC*, i.e., certain symbols are assumed to be associative and/or commutative. Each variant has its own strengths and weaknesses: Kusakari and Toyama [12] introduced a version that results in less dependency pairs than others, but requires a special treatment of AC symbols. Marché and Urbain [13] introduced a version that considers *flattened* terms and results in more dependency pairs than the Kusakari-Toyama version. However, their original version was later pointed out to be incorrect [14]. As a specialization of their more general *equational DP method*, Giesl and Kapur [6] mentioned a version that does not require flattening or a special treatment of AC symbols, so that many techniques for the standard DP framework can be incorporated without major modifications. However, Alarcón et al. [1] pointed out that the minimality property, which is needed for some important techniques such as *usable rules* and the *subterm criterion* [9], does not carry over to the AC case.

* This research was supported by the Austrian Science Fund (FWF) projects Y 757 and P 27502, and the Japan Society for the Promotion of Science (JSPS) KAKENHI #24500012.

© Akihisa Yamada, Christian Sternagel, René Thiemann, and Keiichirou Kusakari;
licensed under Creative Commons License CC-BY

In this paper, we introduce yet another AC-DP framework. Our approach enjoys the strengths of both the Kusakari-Toyama approach and the Giesl-Kapur approach: the number of generated dependency pairs is small and most techniques for the standard DP framework can be integrated without major modifications. Moreover, our approach ensures the minimality property in the standard sense, and thus both the usable rules technique and the subterm criterion become available in our framework.

The key idea of our AC-DP framework is to consider AC axioms just as rewrite rules and to take the dependency pairs of these rules. This choice allows us to reuse most of the reasoning from the standard DP framework, including the notion of minimality.

The results of this paper are formalized using the Isabelle proof assistant [15]. The formalization, consisting of 7069 lines of Isabelle code, is incorporated into our *Isabelle formalization of rewriting* IsaFoR [23], and as it provides a formalized correctness proof of our certifier CeTA, we can now formally validate AC termination proofs. Through experiments CeTA revealed a long-lurking bug in AProVE [5] (in the computation of AC usable rules), which is now fixed. IsaFoR, CeTA, and the experiments are accessible from

`http://cl-informatik.uibk.ac.at/software/ceta/experiments/ac-dp`

The paper is organized as follows. After preliminaries in Section 2, we describe our AC-DP framework in Section 3. In Section 4 we show how to characterize AC termination in our AC-DP framework. We present AC variants of usable rules and the subterm criterion in Sections 5 and 6, respectively. The practical relevance of our work is discussed in Section 7, whereas Section 8 contains a theoretical comparison to related work.

2 Preliminaries

We assume familiarity with term rewriting and only recall some notations required in the remainder; for details on term rewriting, we refer to textbooks [3, 21].

Given a set $\mathcal{F}$ of function symbols with associated arities and a set $\mathcal{V}$ of variables such that $\mathcal{F} \cap \mathcal{V} = \emptyset$, a *term* is either a variable $x \in \mathcal{V}$ or of the form $f(s_1, \ldots, s_n)$, where n is the arity of $f \in \mathcal{F}$ and the arguments $s_1, \ldots, s_n$ are terms. We often abbreviate a list of terms $s_1, \ldots, s_n$ by $\vec{s}_n$. The *root symbol* of a term $s = f(\vec{s}_n)$ is f and denoted by $\mathsf{root}(s)$.

A *substitution* σ assigns each variable x a term $x\sigma$. For a term s, $s\sigma$ denotes the term obtained by replacing every variable x by $x\sigma$ in s. A *position* in a term s is represented by a sequence of natural numbers as usual. The *subterm* of a term s at position p is denoted by $s|_p$, and the term obtained by replacing the subterm by a term t is denoted by $s[t]_p$.

A *rewrite rule* is a pair of terms, written $l \to r$, such that $l \notin \mathcal{V}$ and variables appearing in r also appear in l. A *term rewrite system (TRS)* is a set $\mathcal{R}$ of rewrite rules. There is an $\mathcal{R}$*-rewrite step* from s to t at position p, written $s \xrightarrow[\mathcal{R}]{p} t$, iff there exist a rule $l \to r \in \mathcal{R}$ and a substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. We write $s \xrightarrow[\mathcal{R}]{} t$ if p is irrelevant.

The *associativity* and *commutativity* of a binary symbol $+ \in \mathcal{F}$ are respectively specified by the following axioms, where we use infix notation for $+$:

$$x + (y + z) \approx (x + y) + z \qquad (\mathsf{A}) \qquad\qquad x + y \approx y + x \qquad (\mathsf{C})$$

We assume $\mathcal{E}$ to be a set of associativity and/or commutativity axioms, which we call an *AC theory*. We write $\mathcal{F}_{\mathsf{A}}$ and $\mathcal{F}_{\mathsf{C}}$ for the set of symbols which are associative and commutative in $\mathcal{E}$, respectively, and write $\mathcal{F}_{\mathsf{AC}}$ for $\mathcal{F}_{\mathsf{A}} \cap \mathcal{F}_{\mathsf{C}}$. The equivalence relation induced by $\mathcal{E}$ is denoted by $\underset{\mathcal{E}}{\approx}$.

For a relation R, we denote its transitive closure by R^+ and its reflexive transitive closure by R^*. Given another relation S, we denote the relation $S^* \cdot R \cdot S^*$ by R/S.

For a TRS $\mathcal{R}$ and an AC theory $\mathcal{E}$, $\mathcal{R}$-rewriting modulo $\mathcal{E}$ considers the relation $\xrightarrow[\mathcal{R}/\mathcal{E}]{}$, which is defined as $\xrightarrow[\mathcal{R}]{}/\overset{*}{\underset{\mathcal{E}}{\approx}}$. A term s is *$\mathcal{R}/\mathcal{E}$-nonterminating* iff there exists an infinite sequence $s \xrightarrow[\mathcal{R}/\mathcal{E}]{} s_1 \xrightarrow[\mathcal{R}/\mathcal{E}]{} s_2 \xrightarrow[\mathcal{R}/\mathcal{E}]{} \cdots$, and *$\mathcal{R}/\mathcal{E}$-terminating* otherwise. An $\mathcal{R}/\mathcal{E}$-nonterminating term is *minimal* iff all its proper subterms are $\mathcal{R}/\mathcal{E}$-terminating.

If every term is $\mathcal{R}/\mathcal{E}$-terminating, then we say $\mathcal{R}$ is *terminating modulo* $\mathcal{E}$, or $\mathcal{R}/\mathcal{E}$ is terminating. This terminology carries over to the non-AC case: we say a term is $\mathcal{R}$-terminating, $\mathcal{R}$ is terminating, etc., when the same holds for $\mathcal{R}/\emptyset$.

We extensively use the following notion in subsequent proofs.

▶ **Definition 1** (Well-Founded Order Pair)**.** A *quasi-order pair* on a set A is a pair $\langle \succsim, \succ \rangle$ consisting of a quasi-order $\succsim$ and a transitive relation $\succ$ on A, satisfying $\succ/\succsim \subseteq \succ$. If moreover $\succ$ is well-founded, then we call $\langle \succsim, \succ \rangle$ a *well-founded order pair*.

It is well known that a lexicographic composition of well-founded order pairs again forms a well-founded order pair. Moreover, a well-founded order pair can be extended to a well-founded order pair over multisets [4, 22]. Below $\uplus$ denotes the multiset union.

▶ **Definition 2.** The *multiset extension* of a quasi-order pair $\langle \succsim, \succ \rangle$ on A is the quasi-order pair $\langle \succsim^{\mathsf{mul}}, \succ^{\mathsf{mul}} \rangle$ on multisets over A which is defined as follows: $X \succsim^{\mathsf{mul}} Y$ iff X and Y are of the form $X = \{x_1, \ldots, x_n\} \uplus X'$ and $Y = \{y_1, \ldots, y_n\} \uplus Y'$ such that $\forall i \in \{1, \ldots, n\}.\ x_i \succsim y_i$, and $\forall y \in Y'.\ \exists x \in X'.\ x \succ y$. We have $X \succ^{\mathsf{mul}} Y$ if it also holds that $X' \neq \emptyset$.

3 An AC-Dependency Pair Framework

Before introducing our AC-DP framework, we first recall the basics of dependency pairs. We say a symbol $f \in \mathcal{F}$ is *defined* in $\mathcal{R}$ if there is a rewrite rule $f(\vec{s}_n) \to r \in \mathcal{R}$, and denote the set of defined symbols in $\mathcal{R}$ by $\mathcal{D}_\mathcal{R}$. We assume a fresh *marked* symbol $f^\sharp$ for every $f \in \mathcal{D}_\mathcal{R}$, and write $s^\sharp$ to denote the term $f^\sharp(\vec{s}_n)$ for $s = f(\vec{s}_n)$.

▶ **Definition 3.** A *dependency pair* of a TRS $\mathcal{R}$ is a rule $l^\sharp \to r|_p^\sharp$ such that $l \to r \in \mathcal{R}$ and $\mathsf{root}(r|_p) \in \mathcal{D}_\mathcal{R}$. The set of all dependency pairs of $\mathcal{R}$ is denoted by $\mathsf{DP}(\mathcal{R})$.

The DP framework works on *DP problems*, which are just pairs of TRSs.

▶ **Theorem 4** ([2])**.** *A TRS $\mathcal{R}$ is terminating iff the DP problem $\langle \mathsf{DP}(\mathcal{R}), \mathcal{R} \rangle$ is* finite*, i.e., there is no infinite chain $s_0 \xrightarrow[\mathsf{DP}(\mathcal{R})]{\epsilon} t_0 \xrightarrow[\mathcal{R}]{}^* s_1 \xrightarrow[\mathsf{DP}(\mathcal{R})]{\epsilon} t_1 \xrightarrow[\mathcal{R}]{}^* \cdots$ where every t_i is $\mathcal{R}$-terminating.*

Our key idea towards an AC-DP framework is to represent an AC theory by a (nonterminating) TRS. Commutativity is trivial: axiom (C) can be seen as the rewrite rule $x+y \to y+x$. For associativity, i.e., axiom (A), the following two rules obviously suffice:

$$x + (y + z) \to (x + y) + z \qquad (\mathsf{A}_1) \qquad\qquad (x + y) + z \to x + (y + z) \qquad (\mathsf{A}_2)$$

The benefit of this approach is that now we can take the dependency pairs $\mathsf{DP}(\mathcal{E})$ of AC axioms. Commutativity induces the following dependency pair

$$x +^\sharp y \to y +^\sharp x \qquad (\mathsf{C}^\sharp)$$

and associativity yields the following dependency pairs:

$$x +^\sharp (y + z) \to (x + y) +^\sharp z \qquad (\mathsf{A}^\sharp) \qquad\qquad (x + y) +^\sharp z \to x +^\sharp (y + z) \qquad (\mathsf{A}^\sharp)$$
$$x +^\sharp (y + z) \to x +^\sharp y \qquad (\mathsf{a}^\sharp) \qquad\qquad (x + y) +^\sharp z \to y +^\sharp z \qquad (\mathsf{a}^\sharp)$$

Note that there are other representations; e.g., if $+$ is both associative and commutative, then one of (A_1) and (A_2) may be dropped, since it is a consequence of the other and (C). In the formalization, we only demand that an AC theory $\mathcal{E}$ is represented by a TRS $\mathcal{E}'$ such that $\xrightarrow[\mathcal{E}']{}^* = \mathrel{\underset{\mathcal{E}}{\approx}}$. In the rest of the paper, we identify $\mathcal{E}'$ with $\mathcal{E}$.

▶ **Definition 5.** An *AC-DP problem* is a quadruple of TRSs $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{R}$, and $\mathcal{E}$, denoted by $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$. A $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$*-chain* is a finite or infinite sequence of the form

$$s_0 \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{\epsilon} t_0 \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}^* s_1 \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{\epsilon} t_1 \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}^* \cdots$$

and is called *minimal* if every t_i is $\mathcal{R}/\mathcal{E}$-terminating. An AC-DP problem is said to be *finite* if it admits no minimal chain containing infinitely many $\mathcal{P}$-steps.

We often write $\xrightarrow[\mathcal{P}\cup\mathcal{Q}]{}$ instead of $\xrightarrow[\mathcal{P}\cup\mathcal{Q}]{\epsilon}$ if it is clear from the context that $\mathcal{P}$ and $\mathcal{Q}$ are the first two components of an AC-DP problem.

Our AC-DP problems are quite similar to *relative DP problems* [19, Definition 1], which form the basis of the formalized DP framework in IsaFoR; ignoring minimality, an AC-DP problem $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ corresponds to the relative DP problem $\langle \mathcal{P}, \mathcal{Q}, \emptyset, \mathcal{R} \cup \mathcal{E} \rangle$. Hence all techniques which do not rely on minimality, are immediately applicable.

The most important technique in proving the finiteness of DP problems is the *reduction pair processor* [2, 7]: a reduction pair $\langle \succsim, \succ \rangle$ is a well-founded order pair on terms such that $\succsim$ and $\succ$ are closed under substitutions and $\succsim$ is closed under contexts.

▶ **Theorem 6.** *Let $\langle \succsim, \succ \rangle$ be a reduction pair such that $\mathcal{P} \cup \mathcal{Q} \cup \mathcal{R} \cup \mathcal{E} \subseteq \succsim$. Then, $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ is finite iff $\langle \mathcal{P}'/\mathcal{Q}', \mathcal{R}'/\mathcal{E} \rangle$ is, where $\mathcal{P}' = \mathcal{P} \setminus \succ$, $\mathcal{Q}' = \mathcal{Q} \setminus \succ$, and $\mathcal{R}' = \mathcal{R}$ or $\mathcal{R}' = \mathcal{R} \setminus \succ$ if $\succ$ is closed under contexts.*

The *dependency graph processor* [8, 7] is also easily adapted.

▶ **Theorem 7.** *Let $\mathcal{G}$ be an (estimated) dependency graph, whose set of nodes is $\mathcal{P} \cup \mathcal{Q}$ and there is an arc from $s \to t$ to $u \to v$ whenever there exist substitutions σ and τ such that $t\sigma \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}^* u\tau$. Then $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ is finite iff every AC-DP problem $\langle \mathcal{P} \cap \mathcal{C}_i/\mathcal{Q} \cap \mathcal{C}_i, \mathcal{R}/\mathcal{E} \rangle$ is, where $\mathcal{C}_1, \ldots, \mathcal{C}_n$ are the strongly connected components (SCCs) of $\mathcal{G}$.*

More work has to be done if minimality is involved, since minimality in a relative DP problem $\langle \mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{E} \rangle$ considers termination w.r.t. $\mathcal{R} \cup \mathcal{E}$, whereas we consider termination w.r.t. $\mathcal{R}/\mathcal{E}$ (cf. Sections 5 and 6). But before we can apply any technique, we first have to construct an initial AC-DP problem whose finiteness corresponds to AC termination.

4 From AC Termination to AC-DP Problems

In this section we reduce the termination of a TRS $\mathcal{R}$ modulo an AC theory $\mathcal{E}$ to the finiteness of certain AC-DP problems.

For the standard DP method, i.e., $\mathcal{E} = \mathcal{Q} = \emptyset$, Theorem 4 means that the finiteness of the DP problem is equivalent to the termination of $\mathcal{R}$. In the case of AC termination, however, it is well known that the argument is not directly applicable.

▶ **Example 8** (cf. [12])**.** Consider the TRS $\mathcal{R} = \{ x + x \to \mathsf{a} + \mathsf{b} \}$ with $+ \in \mathcal{F}_{\mathcal{A}}$. The DP problem $\langle \mathsf{DP}(\mathcal{R})/\mathsf{DP}(\mathcal{E}), \mathcal{R}/\mathcal{E} \rangle$ is finite; it is easy to see that $\mathsf{DP}(\mathcal{R}) = \{ x +^\sharp x \to \mathsf{a} +^\sharp \mathsf{b} \}$ cannot constitute an infinite chain. However, $\mathcal{R}/\mathcal{E}$ is nonterminating, as illustrated by

$$(\mathsf{a} + \mathsf{a}) + \mathsf{b} \xrightarrow[\mathcal{R}]{} (\mathsf{a} + \mathsf{b}) + \mathsf{b} \mathrel{\underset{\mathcal{E}}{\approx}} \mathsf{a} + (\mathsf{b} + \mathsf{b}) \xrightarrow[\mathcal{R}]{} \mathsf{a} + (\mathsf{a} + \mathsf{b}) \mathrel{\underset{\mathcal{E}}{\approx}} \cdots$$

Nevertheless, since we take dependency pairs of $\mathcal{E}$, we can reuse the following key lemma from the standard DP method. To be more precise, we will instantiate the lemma for both $\mathcal{R}$ and $\mathcal{E}$ in the later development.

▶ **Lemma 9.** *Let $\mathcal{R}$ be a TRS. If $s \xrightarrow[\mathcal{R}]{\epsilon} t$ for a minimal nonterminating term s and nonterminating t, then there is a minimal nonterminating subterm t' of t with $s^\sharp \xrightarrow[\mathsf{DP}(\mathcal{R})]{\epsilon} t'^\sharp$.*

The following lemma is a variant of [12, Lemma 3.16], but differs in the following way: While [12, Lemma 3.16] requires the so-called *head subterm relation* in a chain to ensure minimality, we employ the rules in $\mathsf{DP}(\mathcal{E})$, especially those of form $(\mathsf{a}^\sharp)$, for this purpose.

▶ **Lemma 10.** *If s is a minimal nonterminating term, then there exist a minimal nonterminating term v and a minimal chain $s^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* \cdot \xrightarrow[\mathcal{R}\cup\mathsf{DP}(\mathcal{R})]{} v^\sharp$.*

The lemma is proved by induction w.r.t. the following well-founded order pair. In the formalization we prove its well-foundedness from the fact that $\mathcal{E}$-equivalent terms have the same size. The same result could be deduced also from a result by Jouannaud and Kirchner [10].

▶ **Definition 11** (Proof Ordering 1)**.** We denote the weak and strict subterm relation by $\trianglerighteq$ and $\triangleright$. The relations $\trianglerighteq_\mathcal{E}$ and $\triangleright_\mathcal{E}$ are defined as $\left(\triangleright \cup \underset{\mathcal{E}}{\approx}\right)^*$ and $\left(\triangleright / \underset{\mathcal{E}}{\approx}\right)^+$, respectively.

Proof of Lemma 10. Consider a minimal nonterminating term s. Then we have a sequence $s \xrightarrow[\mathcal{E}]{}{}^n t \xrightarrow[\mathcal{R}]{} u$ for nonterminating u. We prove the claim by well-founded induction on $\langle s, n\rangle$ w.r.t. the lexicographic composition of $\triangleright_\mathcal{E}$ and $>$ on natural numbers.

- Suppose $s = t$. If $t \xrightarrow[\mathcal{R}]{\epsilon} u$, then by Lemma 9 we obtain a minimal nonterminating term v and a minimal chain $s^\sharp = t^\sharp \xrightarrow[\mathsf{DP}(\mathcal{R})]{} v^\sharp$. Otherwise u must also be minimal and thus by taking $v = u$, we have a minimal chain $s^\sharp = t^\sharp \xrightarrow[\mathcal{R}]{} v^\sharp$.
- Suppose $s \xrightarrow[\mathcal{E}]{} s' \xrightarrow[\mathcal{E}]{}{}^{n-1} t$. Obviously $s^\sharp \xrightarrow[\mathsf{DP}(\mathcal{E})\cup\mathcal{E}]{} s'^\sharp$. Note also that $s \trianglerighteq_\mathcal{E} s'$; hence, if s' is minimal, then the induction hypothesis for $\langle s', n-1\rangle$ yields a minimal nonterminating term v and a minimal chain:

$$s^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{} s'^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* \cdot \xrightarrow[\mathcal{R}\cup\mathsf{DP}(\mathcal{R})]{} v^\sharp$$

  Now suppose s' is not minimal, which is only possible if $s \xrightarrow[\mathcal{E}]{\epsilon} s'$. By Lemma 9, we obtain a minimal nonterminating subterm s'' of s' such that $s^\sharp \xrightarrow[\mathsf{DP}(\mathcal{E})]{} s''^\sharp$. Since s' is not yet minimal, we know $s' \triangleright s''$, and hence $s \triangleright_\mathcal{E} s''$. Since s'' is nonterminating, we have a derivation $s'' \xrightarrow[\mathcal{E}]{}{}^m t' \xrightarrow[\mathcal{R}]{} u'$ for some nonterminating t' and u'. To this sequence, we apply the induction hypothesis for $\langle s'', m\rangle$ and obtain a minimal chain

$$s^\sharp \xrightarrow[\mathsf{DP}(\mathcal{E})]{} s''^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* \cdot \xrightarrow[\mathcal{R}\cup\mathsf{DP}(\mathcal{R})]{} v^\sharp$$

  with a minimal nonterminating term v. ◀

A repeated application of Lemma 10 converts any infinite $\mathcal{R}/\mathcal{E}$-rewrite sequence into an infinite minimal $\langle \mathsf{DP}(\mathcal{R})/\mathsf{DP}(\mathcal{E}), \mathcal{R}/\mathcal{E}\rangle$-chain. However, this AC-DP problem can be *finite*; the resulting chain may have infinitely many $\mathcal{R}$-steps connected by only $\mathsf{DP}(\mathcal{E})$-steps.

We avoid this case via the finiteness of another AC-DP problem. To this end we employ the notion of *AC-extended rewriting* [16].

▶ **Definition 12.** Let x and y be arbitrary fresh variables. The set $\mathcal{R}_\mathcal{E}$ of *extended rules* is the TRS that consists of the following rule for each $l \to r \in \mathcal{R}$ with $\mathsf{root}(l) = + \in \mathcal{F}_{\mathsf{AC}}$

$$l + x \to r + x$$

and the following rules for each $l \to r \in \mathcal{R}$ with $\mathsf{root}(l) = + \in \mathcal{F}_{\mathsf{A}} \setminus \mathcal{F}_{\mathsf{C}}$.

$$x + l \to x + r \qquad l + x \to r + x \qquad x + l + y \to x + r + y$$

We write $\mathcal{R}_\mathcal{E}^\sharp$ for the TRS $\{\, l^\sharp \to r^\sharp \mid l \to r \in \mathcal{R}_\mathcal{E} \,\}$. Note that in this definition, $\mathcal{R}_\mathcal{E}$ does not necessarily contain the rules in $\mathcal{R}$.

Now we state the main theorem of this section.

▶ **Theorem 13.** *A TRS $\mathcal{R}$ is terminating modulo an AC theory $\mathcal{E}$ iff both AC-DP problems $\langle \mathsf{DP}(\mathcal{R})/\mathcal{Q}, \mathcal{R}/\mathcal{E}\rangle$ and $\langle \mathcal{R}_\mathcal{E}^\sharp/\mathcal{Q}, \mathcal{R}/\mathcal{E}\rangle$ are finite, where $\mathcal{Q} = \mathsf{DP}(\mathcal{E})$.*[1]

Before proving the theorem we prepare several notions.

▶ **Definition 14.** A *top position* (also called a *head position* [12]) in an unmarked term $s = f(\vec{s}_n)$ is the root position ϵ if $f \notin \mathcal{F}_{\mathsf{A}}$, or a position p such that $\mathsf{root}(s|_q) = f$ for every prefix q of p, if $f \in \mathcal{F}_{\mathsf{A}}$. The top positions in a marked term $s^\sharp$ are those in s. We denote a rewrite step $s \xrightarrow[\mathcal{R}]{p} t$ as $s \xrightarrow[\mathcal{R}]{\mathsf{top}} t$ if p is a top position in s, and as $s \xrightarrow[\mathcal{R}]{>\mathsf{top}} t$ otherwise.

▶ **Lemma 15.** *If $s \xrightarrow[\mathcal{R}]{\mathsf{top}} t$, then $s \xrightarrow[\mathcal{R}]{\epsilon} t$ or $s \mathrel{\underset{\mathcal{E}}{\approx}} \cdot \xrightarrow[\mathcal{R}_\mathcal{E}]{\epsilon} \cdot \mathrel{\underset{\mathcal{E}}{\approx}} t$.*

As a measure for the inductive proof of the main theorem, we employ *top-flattening* [17]: the top-flattening of a term s w.r.t. $f \in \mathcal{F}_{\mathsf{A}}$ is the multiset $\triangledown_f(s)$ defined inductively by

$$\triangledown_f(s) = \begin{cases} \{s\} & \text{if } \mathsf{root}(s) \neq f \\ \triangledown_f(s_1) \uplus \triangledown_f(s_2) & \text{if } s = f(s_1, s_2) \end{cases}$$

For a term $s = f(\vec{s}_n)$, we define $\triangledown(s)$ as $\triangledown_f(s)$ if $f \in \mathcal{F}_{\mathsf{A}}$, and as $\{s_1, \ldots, s_n\}$ otherwise.

The top-flattenings are ordered by the multiset extension of the following quasi-order pair:

▶ **Definition 16** (Proof Ordering 2)**.** We define the relations $\trianglerighteq_{\mathcal{R}/\mathcal{E}}$ and $\triangleright_{\mathcal{R}/\mathcal{E}}$ as $\trianglerighteq_{\mathcal{R}/\mathcal{E}} := (\xrightarrow[\mathcal{R}]{} \cup \triangleright \cup \mathrel{\underset{\mathcal{E}}{\approx}})^*$ and $\triangleright_{\mathcal{R}/\mathcal{E}} := (\mathrel{\underset{\mathcal{E}}{\approx}} \cdot \xrightarrow[\mathcal{R}]{} \cdot \trianglerighteq \cdot \mathrel{\underset{\mathcal{E}}{\approx}})^+$.

The above relations form a well-founded order pair on $\mathcal{R}/\mathcal{E}$-terminating terms, which is easily shown using the fact that the subterm relation preserves termination. It is also not difficult to prove the following lemma.

▶ **Lemma 17.** *The relations $\trianglerighteq_{\mathcal{R}/\mathcal{E}}$ and $\triangleright_{\mathcal{R}/\mathcal{E}}$ satisfy the following properties:*

1. *If $s^\sharp \xrightarrow[\mathcal{R}]{>\mathsf{top}} t^\sharp$, then $\triangledown(s) \triangleright^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(t)$.*
2. *If $s^\sharp \xrightarrow[\mathcal{E}]{} t^\sharp$, then $\triangledown(s) \trianglerighteq^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(t)$.*

We further require another well-founded order pair for the inductive proof of the main theorem. We write $s \xrightarrow[\mathcal{P}]{\mathsf{min}} t$ when $s \xrightarrow[\mathcal{P}]{\epsilon} t$ *and* t is $\mathcal{R}/\mathcal{E}$-terminating.

[1] Our formalization shows that $\mathsf{DP}(\mathcal{E})$ can be generated w.r.t. only the defined symbols of $\mathcal{R}$. To ease readability, we do not introduce the definition of such restricted dependency pairs.

▶ **Definition 18** (Proof Ordering 3)**.** Let TRSs $\mathcal{Q}$, $\mathcal{R}$, and $\mathcal{E}$ be fixed. For a TRS $\mathcal{P}$, we define the relations $\succsim_{\mathcal{P}}$ and $\succ_{\mathcal{P}}$ by $\left(\xrightarrow[\mathcal{P}\cup\mathcal{Q}]{\mathsf{min}} \cup \xrightarrow[\mathcal{R}\cup\mathcal{E}]{>\epsilon}\right)^*$ and $\left(\xrightarrow[\mathcal{P}]{\mathsf{min}} / \left(\xrightarrow[\mathcal{Q}]{\mathsf{min}} \cup \xrightarrow[\mathcal{R}\cup\mathcal{E}]{>\epsilon}\right)\right)^+$, respectively.

It is easy to see that $\langle \succsim_{\mathcal{P}}, \succ_{\mathcal{P}} \rangle$ forms a quasi-order pair. On the other hand, its well-foundedness depends on the finiteness of the underlying AC-DP problem.

▶ **Lemma 19.** *If $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ is finite, then $\langle \succsim_{\mathcal{P}}, \succ_{\mathcal{P}} \rangle$ is a well-founded order pair on $\mathcal{R}/\mathcal{E}$-terminating terms.*

Proof. We prove the well-foundedness of $\succ_{\mathcal{P}}$ by contradiction: Suppose that it is not well-founded on $\mathcal{R}/\mathcal{E}$-terminating terms. So we have an $\mathcal{R}/\mathcal{E}$-terminating term s that starts an infinite reduction $s \succ_{\mathcal{P}} \cdot \succ_{\mathcal{P}} \cdots$, i.e.,

$$s \left(\xrightarrow[\mathcal{Q}]{\mathsf{min}} \cup \xrightarrow[\mathcal{R}\cup\mathcal{E}]{>\epsilon}\right)^* \cdot \xrightarrow[\mathcal{P}]{\mathsf{min}} \cdot \left(\xrightarrow[\mathcal{Q}]{\mathsf{min}} \cup \xrightarrow[\mathcal{R}\cup\mathcal{E}]{>\epsilon}\right)^* \cdot \xrightarrow[\mathcal{P}]{\mathsf{min}} \cdots$$

This sequence is an infinite minimal chain, contradicting the finiteness of $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$. ◀

Now we are ready to prove the main theorem.

Proof of Theorem 13. The "only if" direction is proved as in the standard DP framework, where we additionally use the fact that $s^\sharp \xrightarrow[\mathcal{R}^\sharp_\mathcal{E}]{} t^\sharp$ implies $s \xrightarrow[\mathcal{R}]{} t$.

For the "if" direction, suppose that $\langle \mathsf{DP}(\mathcal{R})/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ and $\langle \mathcal{R}^\sharp_\mathcal{E}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ are finite. We prove that any minimal $\mathcal{R}/\mathcal{E}$-nonterminating term s derives a contradiction, by induction on $\langle s^\sharp, s^\sharp, \triangledown(s) \rangle$, which is ordered by the lexicographic composition of $\succ_{\mathsf{DP}(\mathcal{R})}$, $\succ_{\mathcal{R}^\sharp_\mathcal{E}}$, and $\rhd^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}}$.

By Lemma 10, we obtain a minimal $\mathcal{R}/\mathcal{E}$-nonterminating term u and a minimal chain $s^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* t^\sharp \xrightarrow[\mathcal{R}\cup\mathsf{DP}(\mathcal{R})]{} u^\sharp$. We distinguish the following cases:

- Suppose $t^\sharp \xrightarrow[\mathsf{DP}(\mathcal{R})]{} u^\sharp$. In this case, we have $s^\sharp \succ_{\mathsf{DP}(\mathcal{R})} u^\sharp$. Hence, the induction hypothesis for u on the first component derives a contradiction.
- Suppose $t^\sharp \xrightarrow[\mathcal{R}]{>\mathsf{top}} u^\sharp$. By definition we have $s^\sharp \succsim_{\mathsf{DP}(\mathcal{R})} u^\sharp$ and $s^\sharp \succsim_{\mathcal{R}^\sharp_\mathcal{E}} u^\sharp$. Using Lemma 17, we also have $\triangledown(s) \trianglerighteq^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(t) \rhd^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(u)$. Hence, the induction hypothesis for u on the third component derives a contradiction.
- Suppose $t^\sharp \xrightarrow[\mathcal{R}]{\mathsf{top}} u^\sharp$. By Lemma 15 we obtain a sequence $t = t_0 \xrightarrow[\mathcal{E}]{} \cdots \xrightarrow[\mathcal{E}]{} t_n \xrightarrow[\mathcal{R}_\mathcal{E}]{\epsilon} v \mathrel{\underset{\mathcal{E}}{\approx}} u$.
  - If every t_i is minimal, then we have

    $$s^\sharp \succsim_{\mathsf{DP}(\mathcal{R})} t^\sharp = t_0^\sharp \succsim_{\mathsf{DP}(\mathcal{R})} t_n^\sharp \succsim_{\mathsf{DP}(\mathcal{R})} v^\sharp$$

    and similarly, $s^\sharp \succsim_{\mathcal{R}^\sharp_\mathcal{E}} t_n^\sharp \succ_{\mathcal{R}^\sharp_\mathcal{E}} v^\sharp$. Thus the induction hypothesis for v on the second component yields a contradiction.
  - Otherwise, for some $i < n$, $t_0, \ldots, t_i$ are minimal but t_{i+1} is not. Take a minimal nonterminating subterm t' of t_{i+1}. Using Lemma 9, we get a minimal chain

    $$s^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* t^\sharp = t_0^\sharp \xrightarrow[\mathcal{E}\cup\mathsf{DP}(\mathcal{E})]{}{}^* t_i^\sharp \xrightarrow[\mathsf{DP}(\mathcal{E})]{} t'^\sharp$$

    and thus, $s^\sharp \succsim_{\mathsf{DP}(\mathcal{R})} t'^\sharp$ and $s^\sharp \succsim_{\mathcal{R}^\sharp_\mathcal{E}} t'^\sharp$. By Lemma 17 we have $\triangledown(s) \trianglerighteq^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(t) \trianglerighteq^{\mathsf{mul}}_{\mathcal{R}/\mathcal{E}} \triangledown(t_i)$, and due to the shape of AC axioms, $\triangledown(t_i) \supset \triangledown(t')$. Hence, the induction hypothesis for t' on the third component derives a contradiction. ◀

In contrast to the non-AC case, Theorem 13 generates two AC-DP problems, where $\mathsf{DP}(\mathcal{R})$ and $\mathcal{R}^\sharp_\mathcal{E}$ are separated. These two sets of pairs can be merged into one problem.

▶ **Corollary 20.** *A TRS $\mathcal{R}$ is terminating modulo an AC theory $\mathcal{E}$ iff the AC-DP problem $\langle (\mathsf{DP}(\mathcal{R}) \cup \mathcal{R}^\sharp_\mathcal{E})/\mathsf{DP}(\mathcal{E}), \mathcal{R}/\mathcal{E} \rangle$ is finite.*

Clearly, solving this merged AC-DP problem cannot be easier than tackling the smaller AC-DP problems separately. Nevertheless, it remains open whether practically there is any difference in power between Theorem 13 and Corollary 20.

5 Usable Rules

In this section, we prove that the usable rules technique is applicable within our AC-DP framework. To this end, we generalize the currently most powerful variant of usable rules in the DP framework [18] to the AC case. To be more precise, we adapt the statements and proofs of the existing formalization in such a way that they become applicable both for the DP framework and the AC-DP framework.

Let us first recapitulate the notions of (non-AC) usable rules. The following function tcap is used to estimate the shape of a term after rewriting:

$$\mathsf{tcap}_{\mathcal{R}}(x) = \text{a fresh variable}$$
$$\mathsf{tcap}_{\mathcal{R}}(f(\vec{s}_n)) = \begin{cases} \text{a fresh variable} & \text{if } \exists l \to r \in \mathcal{R}.\ l \sim f(\mathsf{tcap}_{\mathcal{R}}(\vec{s}_n)) \\ f(\mathsf{tcap}_{\mathcal{R}}(\vec{s}_n)) & \text{otherwise} \end{cases}$$

Here, $\mathsf{tcap}_{\mathcal{R}}(\vec{s}_n)$ is an abbreviation for $\mathsf{tcap}_{\mathcal{R}}(s_1), \ldots, \mathsf{tcap}_{\mathcal{R}}(s_n)$, and $\sim$ denotes *unifiability*: $s \sim t$ iff there exist substitutions σ and τ such that $s\sigma = t\tau$.

Let $\mathcal{R}$ and $\mathcal{U}$ be TRSs. An *argument filter* π assigns each function symbol a subset of its argument positions, where $i \notin \pi(f)$ indicates that the argument s_i of any term $f(\vec{s}_n)$ will be ignored for the usable rules. The formula $\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{S})$ is defined as follows:

$$\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(x) = \mathsf{true}$$
$$\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(f(\vec{s}_n)) = \bigwedge_{i \in \pi(f)} \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(s_i) \wedge \bigwedge_{l \to r \in \mathcal{R}} \big(l \sim f(\mathsf{tcap}_{\mathcal{R}}(\vec{s}_n)) \Longrightarrow l \to r \in \mathcal{U}\big)$$
$$\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{S}) = \bigwedge_{l \to r \in \mathcal{S}} \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(r)$$

Rewrite rules which might be invoked from the right-hand sides of dependency pairs should be considered as usable rules. This is captured formally in the following definition.

▶ **Definition 21** (from [18])**.** We say $\mathcal{U}$ is a set of *usable rules* for a DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ w.r.t. an argument filter π iff the following formula is satisfied:

$$\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{P}) \wedge \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}}(\mathcal{U})$$

The close connection between our AC-DP framework and the standard DP framework admits reusing the above notions without major effort; we simply instantiate $\mathcal{R}$ to $\mathcal{R} \cup \mathcal{E}$ and $\mathcal{P}$ to $\mathcal{P} \cup \mathcal{Q}$. A more precise alternative might integrate $\mathcal{E}$-*unification* into tcap; however, we refrain from this possibility as IsaFoR has no $\mathcal{E}$-unification algorithm formalized yet.

▶ **Definition 22** (Usable Rules for AC)**.** We say $\mathcal{U}$ is a set of usable rules for an AC-DP problem $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ w.r.t. an argument filter π iff the following formula is satisfied:

$$\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}\cup\mathcal{E}}(\mathcal{P} \cup \mathcal{Q}) \wedge \mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}\cup\mathcal{E}}(\mathcal{U})$$

The following theorem generalizes the corresponding theorem in the DP framework [18, Theorem 4.6]. Instead of requiring a weak decrease for all rules in $\mathcal{R} \cup \mathcal{E}$, one just has to look at the usable rules $\mathcal{U}$. Moreover, under certain conditions one can even delete all non-usable rules from $\mathcal{R}$. In the rest of this section, we assume a fresh infix function symbol $\circ$, and denote the TRS $\{\, x \circ y \to x,\ x \circ y \to y \,\}$ by $\mathcal{C}_e$.

▶ **Theorem 23.** *Let $\mathcal{R}$ be a finite TRS, π an argument filter, $\mathcal{U}$ a set of usable rules for $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E}\rangle$ w.r.t. π, and $\langle \succsim, \succ\rangle$ a reduction pair such that*

- *$\mathcal{P} \cup \mathcal{Q} \cup \mathcal{U} \cup \mathcal{C}_e \subseteq \succsim$, and*
- *$\succsim$ is π-compatible; i.e., $f(s_1, \dots, s_i, \dots, s_n) \succsim f(s_1, \dots, s'_i, \dots, s_n)$ whenever $i \notin \pi(f)$.*

Then $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E}\rangle$ is finite if $\langle \mathcal{P}'/\mathcal{Q}', \mathcal{R}'/\mathcal{E}\rangle$ is, where $\mathcal{P}' = \mathcal{P} \setminus \succ$, $\mathcal{Q}' = \mathcal{Q} \setminus \succ$, and $\mathcal{R}' = \mathcal{R}$ or $\mathcal{R}' = (\mathcal{R} \cap \mathcal{U}) \setminus \succ$ if $\mathcal{C}_e \subseteq \succ$ and $\succ$ is closed under contexts.

▶ **Example 24.** To illustrate the application of Theorem 23, consider an AC-DP problem with $\mathcal{F}_{\mathsf{A}} = \{+, \times\}$, $\mathcal{F}_{\mathsf{C}} = \{+, \times, \mathsf{eq}\}$, $\mathcal{P}$ consisting of

$$\mathsf{f}^\sharp(\mathsf{s}(x), y, z) \to \mathsf{eq}^\sharp(x, y) \qquad \mathsf{f}^\sharp(\mathsf{s}(x), y, z) \to \mathsf{f}^\sharp(\mathsf{p}(\mathsf{s}(x)), x + y, x \times z)$$

and $\mathcal{R}$ consisting of a standard rules for addition $+$ and multiplication $\times$, some rules for f, and the following two rules for p.

$$\mathsf{p}(\mathsf{s}(x)) \to x \quad (1) \qquad \mathsf{p}(0) \to 0 \quad (2)$$

If one applies the theorem with $\pi(\mathsf{f}^\sharp) = \{1, 2\}$, then only (1) and the rules for $+$ in $\mathcal{R} \cup \mathcal{E}$ have to be marked as usable. One can ignore

- rules in $\mathcal{R} \cup \mathcal{E}$ for $\times$, since $\times$ occurs only in the third argument of $\mathsf{f}^\sharp$;
- the other rule (2) for p, since its left-hand side does not unify with $\mathsf{p}(\mathsf{s}(x))$; and
- rules in $\mathcal{R} \cup \mathcal{E}$ for f and eq, since only the marked versions of these symbols occur.

It is also necessary to take into account the pairs in $\mathcal{Q}$ for determining usable rules.

▶ **Example 25.** Consider the AC-DP problem $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E}\rangle$ where $\mathcal{E} = \{(\mathsf{A}_1)\}$ and

$$\mathcal{P} = \{\mathsf{f}(x) +^\sharp \mathsf{c} \to \mathsf{a} +^\sharp x\} \quad \mathcal{Q} = \{x +^\sharp (y + z) \to (x + y) +^\sharp z\} \quad \mathcal{R} = \{\mathsf{a} + \mathsf{b} \to \mathsf{f}(\mathsf{b} + \mathsf{c})\}$$

This AC-DP problem is not finite, as illustrated by the following chain:

$$\mathsf{f}(\mathsf{b} + \mathsf{c}) +^\sharp \mathsf{c} \xrightarrow[\mathcal{P}]{} \mathsf{a} +^\sharp (\mathsf{b} + \mathsf{c}) \xrightarrow[\mathcal{Q}]{} (\mathsf{a} + \mathsf{b}) +^\sharp \mathsf{c} \xrightarrow[\mathcal{R}]{} \mathsf{f}(\mathsf{b} + \mathsf{c}) +^\sharp \mathsf{c} \xrightarrow[\mathcal{P}]{} \cdots$$

Now assume one ignores usable rules from $\mathcal{Q}$ – then no rule is usable, and one could wrongly deduce finiteness, e.g., with a polynomial interpretation $\mathcal{A}$ such that

$$x +^\sharp_{\mathcal{A}} y = x +_{\mathcal{A}} y = x + y \qquad \mathsf{f}_{\mathcal{A}}(x) = x \qquad \mathsf{c}_{\mathcal{A}} > \mathsf{a}_{\mathcal{A}}$$

In the non-AC case, a key to proving Theorem 23 is a transformation $\mathcal{I}$, that converts any minimal $\langle \mathcal{P}, \mathcal{R}\rangle$-chain into a $\langle \mathcal{P}, \mathcal{U} \cup \mathcal{C}_e\rangle$-chain by applying $\mathcal{I}$ to all terms in the chain. Here, minimality is essential since $\mathcal{I}$ is applicable only to $\mathcal{R}$-terminating terms.

In the AC-DP framework, we cannot directly reuse $\mathcal{I}$, since minimality ensures only $\mathcal{R}/\mathcal{E}$-termination, but not $\mathcal{R} \cup \mathcal{E}$-termination. Thus we base our soundness proof on the following transformation $\overline{\cdot}$. Below, we write $\overline{\vec{s}_n}$ to denote the list $\overline{s_1}, \dots, \overline{s_n}$.

▶ **Definition 26.** Consider a finite TRS $\mathcal{R}$, an AC theory $\mathcal{E}$, and a set $\mathcal{U}$ of usable rules. For an $\mathcal{R}/\mathcal{E}$-terminating term s, we define $\overline{s}$ as follows: $\overline{x} = x$ and for $s = f(\vec{s}_n)$,

$$\overline{s} = \begin{cases} \mathsf{list}\big(\{g(\overline{\vec{t}_n}) \mid s \approx_{\mathcal{E}} g(\vec{t}_n)\} \cup \{\overline{t} \mid s \xrightarrow[\mathcal{R}/\mathcal{E}]{} t\}\big) & \text{if } \exists l \to r \in (\mathcal{R} \cup \mathcal{E}) \setminus \mathcal{U}.\ l \sim f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{s}_n)) \\ f(\overline{\vec{s}_n}) & \text{otherwise} \end{cases}$$

Here, $\mathsf{list}(\{s_1, \dots, s_n\})$ denotes the term $s_1 \circ \cdots \circ s_n$, where the elements are sorted w.r.t. an arbitrary but fixed total order on terms.

It is easy to see that $\overline{s}$ is defined for every $\mathcal{R}/\mathcal{E}$-terminating s, using an inductive argument and the fact that there are only finitely many $\mathcal{E}$-equivalent terms.

Most of the existing proofs using transformation $\mathcal{I}$ can be immediately generalized to the new transformation; various properties [18, Lemma 4.9] are satisfied after straightforward modifications, e.g., replacing $\mathcal{R}$-termination by $\mathcal{R}/\mathcal{E}$-termination.

Nevertheless, in order to prove Theorem 23 we have to add some new properties which are required for simulating $\mathcal{E}$-equivalence. In the following crucial property we rely upon the fact that $\mathcal{E}$ is an AC theory; in many other places it was sufficient to know that $\mathcal{E}$ is symmetric and has only finite equivalence classes. We omit presenting other details of the proof of Theorem 23.

▶ **Lemma 27.** *Suppose that $\mathcal{E}$ is an AC theory and $\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}\cup\mathcal{E}}(\mathcal{U})$ is satisfied. If $f(\vec{t}_n)$ is $\mathcal{R}/\mathcal{E}$-terminating, $f(\vec{t}_n) \approx_{\mathcal{E}} s$, and $\overline{f(\vec{t}_n)} \neq f(\overline{\vec{t}_n})$, then $\overline{f(\vec{t}_n)} = \overline{s}$.*

Proof. By the preconditions s must be of the form $f(\vec{s}_n)$, and due to the symmetry of $\approx_{\mathcal{E}}$, $f(\vec{s}_n) \approx_{\mathcal{E}} f(\vec{t}_n)$. Furthermore, there must be a rule $l \to r \in (\mathcal{R} \cup \mathcal{E}) \setminus \mathcal{U}$ such that $l \sim f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{t}_n))$. In particular, $\mathsf{root}(l)$ must be f.

The challenge is showing $\overline{f(\vec{s}_n)} \neq f(\overline{\vec{s}_n})$, i.e., finding a non-usable rule where the left-hand side unifies with $f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{s}_n))$. Having proved this fact, the claim immediately follows by the transitivity of $\approx_{\mathcal{E}}$ and the definition of $\overline{\cdot}$.

To determine such a non-usable rule we consider the following two cases.

- If $s_i \approx_{\mathcal{E}} t_i$ for all $1 \leq i \leq n$, then $f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{s}_n)) = f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{t}_n))$. Hence, the above $l \to r$ is a desired non-usable rule.
- Otherwise, we have a root $\mathcal{E}$-step, i.e., for some AC rule $u \to v \in \mathcal{E}$, $f(\vec{s}_n) \approx_{\mathcal{E}} f(\vec{w}_n) = u\sigma$ and $v\sigma \approx_{\mathcal{E}} f(\vec{t}_n)$. The properties of tcap guarantee $u \sim f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{s}_n))$. By the shape of the AC rules and $\mathsf{urClosed}^{\pi}_{\mathcal{U},\mathcal{R}\cup\mathcal{E}}(\mathcal{U})$, we conclude $u \to v \notin \mathcal{U}$ from the fact that $l \to r \notin \mathcal{U}$ and $\mathsf{root}(l) = f$. Hence $u \to v \in (\mathcal{R}\cup\mathcal{E})\setminus\mathcal{U}$, and in combination with $u \sim f(\mathsf{tcap}_{\mathcal{R}\cup\mathcal{E}}(\vec{s}_n))$ we have found the desired non-usable rule. ◀

6 Subterm Criterion

The subterm criterion [9] is an efficient technique for proving termination in the standard DP framework. It is based on the notion of *simple projections*, i.e., a mapping π that assigns to each marked symbol $f^\sharp$ one of its argument positions $\pi(f^\sharp)$. Simple projections are extended to terms as follows: $\pi(f^\sharp(s_1,\ldots,s_n)) = s_i$ for $i = \pi(f^\sharp)$. For a relation $\sqsupset$ on terms, we denote $\pi(s) \sqsupset \pi(t)$ by $s \sqsupset^\pi t$.

▶ **Theorem 28** ([9])**.** *Let $\langle\mathcal{P},\mathcal{R}\rangle$ be a (standard) DP problem, and π a simple projection such that $\mathcal{P} \subseteq \trianglerighteq^\pi$. Then, $\langle\mathcal{P},\mathcal{R}\rangle$ is finite iff $\langle\mathcal{P} \setminus \triangleright^\pi, \mathcal{R}\rangle$ is.*

This technique however is not directly applicable if AC symbols appear. For commutative symbols, neither of the argument positions can be projected.

▶ **Example 29.** For the TRS $\{\,\mathsf{s}(x)+\mathsf{p}(y) \to x+y\,\}$ with $+ \in \mathcal{F}_{\mathsf{C}}$ we construct the following pair for the TRS and in addition the pair $(\mathsf{C}^\sharp)$ for the equations.

$$\mathsf{s}(x) +^\sharp \mathsf{p}(y) \to x +^\sharp y \tag{3}$$

We would like to delete (3) by the subterm criterion. However, projecting either argument of $+^\sharp$, $(\mathsf{C}^\sharp)$ cannot be oriented by $\trianglerighteq^\pi$. On the other hand, one cannot ignore $(\mathsf{C}^\sharp)$; e.g., consider removing a pair $\mathsf{s}(x) +^\sharp x \to x +^\sharp \mathsf{s}(x)$ via $\pi(+^\sharp) = 1$.

This motivates us to generalize the range of π to *multisets of arguments*. For instance, in Example 29 one can choose $\pi(+^\sharp) = \{1, 2\}$. Then $(3) \in \rhd^\pi$ and $(\mathsf{C}^\sharp) \subseteq \trianglerighteq^\pi$.

Associativity rules also require a careful treatment. As in the commutative case, ignoring $(\mathsf{A}^\sharp)$ is unsound, but $(\mathsf{A}^\sharp) \subseteq \trianglerighteq^\pi$ is impossible.

▶ **Example 30** (Example 29 continued)**.** If we assume that $+ \in \mathcal{F}_{\mathsf{AC}}$, the AC-DP problem will contain dependency pairs from $(\mathsf{A}^\sharp)$ and $(\mathsf{a}^\sharp)$. For $\pi(+^\sharp) = \{1, 2\}$, the projected left- and right-hand sides of $(\mathsf{A}^\sharp)$ are $\{x, y + z\}$ and $\{x + y, z\}$, which are incomparable.

This motivates us to also allow projections for unmarked symbols, e.g., $+$ in the above example, and perform the projection recursively.

▶ **Definition 31.** A *multi-projection* π for a set $\mathcal{G}$ of symbols is a mapping that assigns every symbol $f \in \mathcal{G}$ a non-empty multiset $\pi(f)$ of its argument positions. From π we define a mapping from terms to multisets of terms, also denoted by π, as follows:

$$\pi(s) = \begin{cases} \pi(s_{i_1}) \uplus \cdots \uplus \pi(s_{i_m}) & \text{if } s = f(\vec{s}_n),\ f \in \mathcal{G} \text{ and } \pi(f) = \{i_1, \ldots, i_m\} \\ \{s\} & \text{otherwise} \end{cases}$$

For a relation $\sqsupset$ on terms, we write $s \sqsupset^\pi t$ for $\pi(s) \sqsupset^{\mathsf{mul}} \pi(t)$.

Now we can satisfy the constraints in Example 30: $\pi(+^\sharp) = \pi(+) = \{1, 2\}$ will ensure $\{(\mathsf{C}^\sharp), (\mathsf{A}^\sharp)\} \subseteq \trianglerighteq^\pi$ and $\{(3), (\mathsf{a}^\sharp)\} \subseteq \rhd^\pi$. However, allowing projections for a defined symbol – no matter whether it is an AC symbol or not – requires a further side-condition.

▶ **Example 32.** Consider the TRS $\mathcal{R} = \{\, \mathsf{f}(\mathsf{s}(x)) \to \mathsf{f}(\mathsf{g}(x)), \mathsf{g}(x) \to \mathsf{s}(x) \,\}$ and the multi-projection $\pi(\mathsf{f}^\sharp) = \pi(\mathsf{g}^\sharp) = \pi(\mathsf{g}) = \{1\}$. We have $\mathsf{DP}(\mathcal{R}) = \{\, \mathsf{f}^\sharp(\mathsf{s}(x)) \to \mathsf{f}^\sharp(\mathsf{g}(x)) \,\} \subseteq \rhd^\pi$, but the DP problem $\langle \mathsf{DP}(\mathcal{R}), \mathcal{R} \rangle$ is not finite.

Thus in the following theorem, we demand that whenever $\mathcal{G}$ contains a defined symbol f, the rules that define f should be oriented by $\trianglerighteq^\pi$.

▶ **Theorem 33.** *Let $\mathcal{E}$ be a size preserving TRS and π a multi-projection over $\mathcal{G}$ such that $\mathcal{P} \cup \mathcal{Q} \subseteq \trianglerighteq^\pi$ and $l \trianglerighteq^\pi r$ for every $l \to r \in \mathcal{R} \cup \mathcal{E}$ with $\mathsf{root}(l) \in \mathcal{G}$. Then $\langle \mathcal{P}/\mathcal{Q}, \mathcal{R}/\mathcal{E} \rangle$ is finite iff $\langle \mathcal{P}'/\mathcal{Q}', \mathcal{R}/\mathcal{E} \rangle$ is, where $\mathcal{P}' = \mathcal{P} \setminus \rhd^\pi$ and $\mathcal{Q}' = \mathcal{Q} \setminus \rhd^\pi$.*

In order to prove the theorem, we need to prove that both $\rhd^\pi$ and $\trianglerighteq^\pi$ are closed under substitutions – a trivial property of $\rhd$ and $\trianglerighteq$.

▶ **Lemma 34.** *If $s \trianglerighteq^\pi t$ then $s\sigma \trianglerighteq^\pi t\sigma$. If $s \rhd^\pi t$ then $s\sigma \rhd^\pi t\sigma$.*

The proof of the lemma is not so trivial, since substitutions may affect the recursive application of π. Moreover, the restriction in Definition 31 that $\pi(f)$ is nonempty for every $f \in \mathcal{G}$ is crucial for $\rhd^\pi$ to be closed under substitutions.

▶ **Example 35.** Let $\pi(+) = \{1, 2\}$. We have $\mathsf{a} + x \rhd^\pi \mathsf{a}$ since $\{\mathsf{a}, x\} \rhd^{\mathsf{mul}} \{\mathsf{a}\}$. Consider substituting x by $\mathsf{f}(\mathsf{a})$, and allowing $\pi(\mathsf{f}) = \emptyset$. Then $\mathsf{a} + \mathsf{f}(\mathsf{a}) \rhd^\pi \mathsf{a}$ does not hold, since $\{\mathsf{a}\} \rhd^{\mathsf{mul}} \{\mathsf{a}\}$ does not hold.

Proof of Theorem 33. Let $\mathcal{S} = \mathcal{P} \setminus \mathcal{P}' \cup \mathcal{Q} \setminus \mathcal{Q}'$. We only prove that there is no minimal chain containing infinitely many $\mathcal{S}$-steps; the remaining reasoning is trivial. So, assume to the contrary that there is a minimal chain with infinitely many $\mathcal{S}$-steps:

$$s_0 \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{} t_0 \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}{}^{*} s_1 \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{} t_1 \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}{}^{*} s_2 \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{} \cdots$$

Table 1 Experiments.

	AProVE						NaTT					
	certified		full		no ACDP		Thm. 13		Cor. 20		full	
	#	time	#	time	#	time	#	time	#	time	#	time
yes	128	560.3	128	508.5	82	166.3	78	6.2	78	6.0	113	86.3
no	0	–	2	21.2	0	–	0	–	0	–	1	0.5
maybe	14	310.6	12	340.6	63	189.7	67	46.3	67	28.4	31	149.2
timeout	3	1080.0	3	1080.0	0	–	0	–	0	–	0	–

Let $\langle \succsim, \succ \rangle$ be the quasi-order pair defined as $\succsim := \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}^*$ and $\succ := (\rhd / \xrightarrow[\mathcal{R}\cup\mathcal{E}]{})^+$. By the preconditions and Lemma 34 we can turn every step $s_i \xrightarrow[\mathcal{P}\cup\mathcal{Q}]{} t_i$ into $\pi(s_i) \trianglerighteq^{\mathsf{mul}} \pi(t_i)$, and whenever $s_i \xrightarrow[\mathcal{S}]{}^* t_i$ then $\pi(s_i) \rhd^{\mathsf{mul}} \pi(t_i)$. Moreover, from $t_i \xrightarrow[\mathcal{R}\cup\mathcal{E}]{}^* s_{i+1}$ we derive $\pi(t_i) \succsim^{\mathsf{mul}} \pi(s_{i+1})$ by the conditions on the rules of $\mathcal{R}\cup\mathcal{E}$. Hence, by using $\rhd \subseteq \succ$ we obtain

$$\pi(s_0) \succsim^{\mathsf{mul}} \pi(t_0) \succsim^{\mathsf{mul}} \pi(s_1) \succsim^{\mathsf{mul}} \pi(t_1) \succsim^{\mathsf{mul}} \pi(s_2) \succsim^{\mathsf{mul}} \dots$$

with an infinite number of $\succ^{\mathsf{mul}}$ steps. Thus, some element $u \in \pi(t_0)$ must start an infinite derivation w.r.t. $\succ$. The size preservation of $\mathcal{E}$ implies that this derivation from u must contain infinitely many $\mathcal{R}$-steps; i.e., u is not $\mathcal{R}/\mathcal{E}$ terminating. This contradicts the minimality of the chain, since u is a subterm of t_0, which is $\mathcal{R}/\mathcal{E}$ terminating. ◀

Theorem 33 generalizes the subterm criterion even in the standard case; one may freely include non-defined symbols into $\mathcal{G}$, which was previously allowed only for marked defined symbols. A similar generalization has been proposed for higher-order rewriting [11]. Nevertheless, in case marked associative symbols are involved, Theorem 33 may not work as one expects from a subterm criterion, as shown in the following example.

▶ **Example 36.** Consider the TRS $\mathcal{R} = \{ \mathsf{s}(x) + y \to \mathsf{s}(x + y) \}$ with $+ \in \mathcal{F}_{\mathsf{A}}$. We construct the following dependency pair for $\mathcal{R}$, in addition to $(\mathsf{A}^\sharp)$ and $(\mathsf{a}^\sharp)$:

$$\mathsf{s}(x) +^\sharp y \to x +^\sharp y \tag{4}$$

One might expect deleting (4) by the subterm criterion; however, $(\mathsf{A}^\sharp)$ demands $+ \in \mathcal{G}$ and thus $\mathcal{R} \subseteq \trianglerighteq^\pi$. This is possible only if s is also in $\mathcal{G}$, resulting $(4) \in \trianglerighteq^\pi$ but $(4) \notin \rhd^\pi$. Hence, on this example other techniques have to be applied, such as reduction pairs with usable rules as in Theorem 23.

7 Experiments

We extended the *certification problem format (CPF)* [20] to our AC-DP framework. For proving finiteness of AC-DP problems, all techniques of Sections 3 to 6 are supported. As a benchmark we use 145 AC termination problems from various sources. Further details on the experimental setup is available on the accompanying website.

We adjusted the termination prover AProVE, that already implements an (unpublished) AC-DP framework based on the Giesl-Kapur variant, to comply to our new framework. We also implemented our technique in NaTT [24], where both Theorem 13 and Corollary 20 are available for comparison. Although any reduction pair supported by CeTA can be used also for AC termination, we disabled the *recursive path order* (RPO) and *Knuth-Bendix order*

(KBO) in the termination provers, since CeTA does not support their AC-compatible variants AC-RPO [17] and AC-KBO [25].

We tested three configurations of AProVE: the new "certified" mode, "full" mode that uses all (uncertified) techniques implemented in AProVE, and "no ACDP" mode that is limited to direct techniques that do not require dependency pairs, which is supported also by an earlier version of CeTA that does not include the AC-DP framework.

The result is shown in the left half of Table 1, where runtime is measured in seconds. Our AC-DP framework is clearly more powerful than "no ACDP" mode. Between "full" and "certified" modes we observe no difference in power; for all examples where "full" mode applied AC-RPO, there are alternative proofs via the AC-DP framework and non-linear polynomial interpretations in "certified" mode. In earlier experiments CeTA rejected one proof generated by a previous version of AProVE, where the usable rules computation in the presence of commutative symbols was incorrect. The bug is now corrected – our work indeed improved the reliability of a state-of-the-art termination prover.

The results for NaTT are shown in the right half of Table 1. Since non-linear polynomials are not supported in NaTT, its "full" strategy is significantly better than the certifiable strategies. For this difference, AC-RPO plays a crucial role.

In one example (`RENAMED-BOOL_complete-noand`), NaTT with Theorem 13 took 21.0sec before it eventually gave up, while with Corollary 20 it gave up after only 0.6sec. This however does not indicate that Theorem 13 is weaker or stronger than Corollary 20; the latter configuration quickly failed due to a small SCC that stems from $\mathsf{DP}(\mathcal{E})$, which would be handled later as another AC-DP problem in the former configuration.

8 Related Work

Kusakari and Toyama [12] introduced a term marking $\cdot^{\#}$ that treats AC symbols specially. If the root symbol of s is an AC symbol $+$, then $s^{\#}$ puts marks on every $+$ occurring at a top position. For instance, $\big(\mathsf{a} + (x + \mathsf{s}(y + z))\big)^{\#} = \mathsf{a} +^{\sharp} (x +^{\sharp} \mathsf{s}(y + z))$. Note that rewriting in such a deeply marked term may bring unmarked AC symbols to a top position; consider, e.g., rewriting $\mathsf{a} \to \mathsf{b} + \mathsf{b}$ in the above term. Therefore, a chain must implicitly perform the following rewriting (AC-mark) and, in order to maintain minimality, also (AC-del).

$$(x + y) +^{\sharp} z \leftrightarrow (x +^{\sharp} y) +^{\sharp} z \quad \text{(AC-mark)} \qquad (x +^{\sharp} y) +^{\sharp} z \to x +^{\sharp} y \quad \text{(AC-del)}$$

The rule (AC-del) is similar but not equal to ($\mathsf{a}^{\sharp}$); besides the nested marks, notice the difference in the right-hand sides.

The special behavior of marking causes major difficulties with respect to implementation and formalization. We also tried but did not succeed to prove usable rules for the Kusakari-Toyama formulation. Fortunately, such an effort is actually not necessary, since our formulation subsumes the Kusakari-Toyama version, in the following sense:

▶ **Proposition 37.** *If a TRS is shown to be AC terminating by the AC-DPs of Kusakari-Toyama with some reduction pairs, then our version succeeds with the same reduction pairs.*

Proof. Consider an AC-compatible reduction pair $\langle \succsim, \succ \rangle$ weakly orienting (AC-mark) and (AC-del). Whenever $s^{\#} \mathrel{(\succsim_{/\sim})} t^{\#}$ we have $s^{\sharp} \mathrel{(\succsim_{/\sim})} t^{\sharp}$, because of (AC-mark). Also ($\mathsf{a}^{\sharp}$) is at least weakly oriented due to (AC-mark), (AC-del), and the fact that $+^{\sharp}$ is assumed to be associative in their formulation. ◀

The converse does not hold if estimated dependency graph techniques are allowed.

▶ **Example 38.** Consider the TRS $\mathcal{R}$ with the following three rules:

$$(\mathsf{a}+\mathsf{a})+x \to (\mathsf{a}+\mathsf{b})+x \qquad \mathsf{b}+\mathsf{c} \to \mathsf{d}(\mathsf{a}+\mathsf{c}) \qquad \mathsf{c} \to \mathsf{a}+\mathsf{a}$$

where $+ \in \mathcal{F}_{\mathsf{A}}$. The dependency pairs of $\mathcal{R}$ in terms of [12] are

$$(\mathsf{a}+^\sharp \mathsf{a})+^\sharp x \to (\mathsf{a}+^\sharp \mathsf{b})+^\sharp x \quad (5) \qquad \mathsf{b}+^\sharp \mathsf{c} \to \mathsf{a}+^\sharp \mathsf{c} \quad (7) \qquad \mathsf{c}^\sharp \to \mathsf{a}+^\sharp \mathsf{a} \quad (9)$$
$$(\mathsf{a}+^\sharp \mathsf{a})+^\sharp x \to \mathsf{a}+^\sharp \mathsf{b} \quad (6) \qquad \mathsf{b}+^\sharp \mathsf{c} \to \mathsf{c}^\sharp \quad (8)$$

and the extended rules are

$$((\mathsf{a}+^\sharp \mathsf{a})+^\sharp x)+^\sharp z \to ((\mathsf{a}+^\sharp \mathsf{b})+^\sharp x)+^\sharp z \quad (10) \qquad (\mathsf{b}+^\sharp \mathsf{c})+^\sharp z \to \mathsf{d}(\mathsf{a}+\mathsf{c})+^\sharp z \quad (11)$$

A polynomial interpretation with $\mathsf{c}_{\mathcal{A}} = \mathsf{c}^\sharp_{\mathcal{A}} = 1$ and $\mathsf{a}_{\mathcal{A}} = 0$ would easily remove (9), and afterwards (8) does not constitute an SCC. The extended rule (11) would also be easy.

The remaining SCC is $\{(5), (6), (7), (10)\}$, where no AC-reduction pair technique we know can be applied. Note that in the Kusakari-Toyama formulation, the dependency graph has an edge from (5) to (7), and there is clearly one for the other direction.

In our formulation, the corresponding SCC (ignoring the nested marks) is $\{(5), (6), (7), (\mathsf{a}^\sharp), (\mathsf{A}^\sharp)\}$. From here $(\mathsf{a}^\sharp)$ can be removed, e.g., by a polynomial interpretation with $x +_{\mathcal{A}} y = x + y + 1$. Then (7) does not constitute an SCC, and remaining $\{(5), (6), (\mathsf{A}^\sharp)\}$ is easy. The AC-DP problem for the extended rules is also easy. Hence our formulation proves the termination of $\mathcal{R}/\mathcal{E}$, while the Kusakari-Toyama formulation fails.

Marché and Urbain [13] considered *flattened* terms, which would require some extra formalization work. However, their original version (with marks) was reported to be unsound [14], since it lacks a rule that corresponds to (AC-mark). They also do not impose a counterpart of our $(\mathsf{a}^\sharp)$ or (AC-del) of Kusakari-Toyama; the price for this omission is that the minimality of a chain becomes nontrivial to define [1]. It is indeed unclear, e.g., how to adapt the usable rules technique; cf. Example 25. Apart from that, their formulation takes $\mathsf{DP}(\mathcal{R} \cup \mathcal{R}_{\mathcal{E}})$, which is in general a superset of $\mathsf{DP}(\mathcal{R})$ and $\mathcal{R}^\sharp_{\mathcal{E}}$ and often imposes more dependency pairs.

Giesl and Kapur [6] considered a set $\mathcal{E}$ of more general equations than AC, and regarded $\mathsf{DP}(\mathsf{Inst}_{\mathcal{E}}(\mathcal{R} \cup \mathcal{R}_{\mathcal{E}}))$ together with $\mathcal{E}^\sharp$, that is, $(\mathsf{C}^\sharp)$ and $(\mathsf{A}^\sharp)$ for AC. They further suggested that for the AC case, taking $\mathsf{Inst}_{\mathcal{E}}$ is not necessary and hence $\mathsf{DP}(\mathcal{R} \cup \mathcal{R}_{\mathcal{E}})$ suffices. For this simplification, however, they remarked that the notion of minimality has to be modified [6].

Below we elaborate more on the problem, using an example inspired by Alarcón et al. [1].

▶ **Example 39.** Consider the TRS $\mathcal{R}$ consisting of the following three rules:

$$\mathsf{a}\cdot\mathsf{a} \to \mathsf{a}\cdot\mathsf{b}\cdot\mathsf{c} \qquad \mathsf{b}\cdot\mathsf{b} \to \mathsf{a}\cdot\mathsf{b}\cdot\mathsf{c} \qquad \mathsf{c}\cdot\mathsf{c} \to \mathsf{a}\cdot\mathsf{b}\cdot\mathsf{c}$$

where $\cdot \in \mathcal{F}_{\mathsf{AC}}$. We abbreviate $s \cdot t$ by st. Note that all subterms of abc, the common right-hand side in $\mathcal{R}$, are terminating. Hence, the only dependency pairs in $\mathsf{DP}(\mathcal{R} \cup \mathcal{R}_{\mathcal{E}})$ that constitute an infinite chain are

$$\mathsf{aa} \cdot^\sharp x \to \mathsf{abc} \cdot^\sharp x \qquad \mathsf{bb} \cdot^\sharp x \to \mathsf{abc} \cdot^\sharp x \qquad \mathsf{cc} \cdot^\sharp x \to \mathsf{abc} \cdot^\sharp x$$

The TRS $\mathcal{R}$ is not terminating modulo $\mathcal{E}$; there are essentially six (modulo $\mathcal{E}$) minimal nonterminating terms, namely aab, aac, abb, bbc, acc, and bcc. Take, e.g., $s = \mathsf{aab}$. Any infinite chain from $s^\sharp$ has the following prefix:

$$\mathsf{aa} \cdot^\sharp \mathsf{b} \to \mathsf{abc} \cdot^\sharp \mathsf{b} \underset{\mathcal{E}\cup\mathcal{E}^\sharp}{\approx} \mathsf{bb} \cdot^\sharp \mathsf{ac} \to \mathsf{abc} \cdot^\sharp \mathsf{ac}$$

From here, there are the following two ways to continue the chain:

- $\mathsf{abc} \cdot^\sharp \mathsf{ac} \underset{\mathcal{E}\cup\mathcal{E}^\sharp}{\approx} \mathsf{aa} \cdot^\sharp \mathsf{bcc} \to \mathsf{abc} \cdot^\sharp \mathsf{bcc} = t^\sharp$, or
- $\mathsf{abc} \cdot^\sharp \mathsf{ac} \underset{\mathcal{E}\cup\mathcal{E}^\sharp}{\approx} \mathsf{cc} \cdot^\sharp \mathsf{aab} \to \mathsf{abc} \cdot^\sharp \mathsf{aab} = u^\sharp$.

Both $t^\sharp$ and $u^\sharp$ contain a nonterminating subterm bcc or aab. Due to symmetry, we cannot have an infinite chain that satisfies the minimality in terms of Giesl and Kapur [6].

Alarcón et al. [1] proposed the notion of A∨C theories, which we simply call AC theories in this paper. In order to define a suitable notion of minimality, they introduced the notion of *stable minimal* terms: any subterm of any AC-equivalent of such a term should be terminating. To maintain stable minimality in a chain, they re-invented (AC-del) without nested marks:

$$(x + y) +^\sharp z \to x +^\sharp y \tag{12}$$

Note that this is still different from our ($\mathsf{a}^\sharp$), and the difference is crucial: (12) is introduced to extract a nonterminating subterm in a chain (an instance of $x + y$) into the main component of the chain (as $x +^\sharp y$). Thus a chain must admit nonterminating subterms, which disallow adapting the proofs of usable rules.[2] Besides, they choose $\mathsf{DP}(\mathcal{R} \cup \mathcal{R}_\mathcal{E})$ and $\mathcal{E}^\sharp$ following Giesl and Kapur. As in Proposition 37, our approach subsumes also this formulation.

9 Conclusion

We have formalized an AC dependency pair framework for proving termination modulo AC axioms. We extended techniques such as dependency graph estimations, reduction pairs with usable rules, and the subterm criterion, for proving AC termination.

A formalization of reduction pairs like AC-RPO and experimental evaluation of the AC subterm criterion is left as future work. Moreover, a formalized algorithm for $\mathcal{E}$-unification would enable more precise estimations of dependency graphs and usable rules.

Acknowledgments. We would like to thank Aart Middeldorp for helpful discussions, and the anonymous referees for their helpful feedback. The benchmark problems were collected by Sarah Winkler.

References

1 B. Alarcón, S. Lucas, and J. Meseguer. A dependency pair framework for A∨C-termination. In *WRLA 2010*, volume 6381 of *LNCS*, pages 36–52, 2010.
2 T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Compt. Sci.*, 236(1-2):133–178, 2000.
3 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
4 N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.
5 J. Giesl, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, C. Otto, M. Plucker, P. Schneider-Kamp, T. Stroder, S. Swiderski, and R. Thiemann. Proving termination of programs automatically with AProVE. In *IJCAR 2014*, LNAI 8562, pages 184–191, 2014.
6 J. Giesl and D. Kapur. Dependency pairs for equational rewriting. In *RTA 2001*, volume 2051 of *LNCS*, pages 93–107, 2001.

[2] Their unpublished report contains a proof of usable rules, which however incorrectly assumes that a chain contains only terminating terms.

7 J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *LPAR 2004*, volume 3452 of *LNAI*, pages 75–90, 2004.

8 N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Inf. Comput.*, 199(1,2):172–199, 2005.

9 N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool: Techniques and features. *Inf. Comput.*, 205(4):474–511, 2007.

10 J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986.

11 K. Kusakari and M. Sakai. Enhancing dependency pair method using strong computability in simply-typed term rewriting. *Appl. Algebr. Eng. Comm. Compt.*, 18(5):407–431, 2007.

12 K. Kusakari and Y. Toyama. On proving AC-termination by AC-dependency pairs. *IEICE T. Inf. Syst.*, E84-D(5):439–447, 2001.

13 C. Marché and X. Urbain. Termination of associative-commutative rewriting by dependency pairs. In *RTA 1998*, volume 1379 of *LNCS*, pages 241–255, 1998.

14 C. Marché and X. Urbain. Modular and incremental proofs of AC-termination. *J. Symb. Comput.*, 38(1):873–897, 2004.

15 T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

16 G. Peterson and M. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, 1981.

17 A. Rubio. A fully syntactic AC-RPO. *Inf. Comput.*, 178(2):515–533, 2002.

18 C. Sternagel and R. Thiemann. Certified subterm criterion and certified usable rules. In *RTA 2010*, volume 6 of *LIPIcs*, pages 325–340, 2010.

19 C. Sternagel and R. Thiemann. A relative dependency pair framework. In *WST 2012*, pages 79–83, 2012.

20 Christian Sternagel and Rene Thiemann. The certification problem format. In *UITP 2014*, volume 167 of *EPTCS*, pages 61–72, 2014.

21 TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

22 R. Thiemann, G. Allais, and J. Nagele. On the formalization of termination techniques based on multiset orderings. In *RTA 2012*, volume 15 of *LIPIcs*, pages 339–354, 2012.

23 R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *TPHOLs 2009*, volume 5674 of *LNCS*, pages 452–468, 2009.

24 A. Yamada, K. Kusakari, and T. Sakabe. Nagoya Termination Tool. In *RTA-TLCA 2014*, volume 8560 of *LNCS*, pages 466–475, 2014.

25 A. Yamada, S. Winkler, N. Hirokawa, and A. Middeldorp. AC-KBO revisited. *Theor. Pract. Log. Prog.*, 16(2):163–188, 2014.

The Directed Homotopy Hypothesis

Jérémy Dubut[1], Éric Goubault[2], and Jean Goubault-Larrecq[3]

1 **LSV, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France; and LIX, Ecole Polytechnique, CNRS, Université Paris-Saclay, 91128 Palaiseau, France**
dubut@lsv.ens-cachan.fr

2 **LIX, Ecole Polytechnique, CNRS, Université Paris-Saclay, 91128 Palaiseau, France**
goubault@lix.polytechnique.fr

3 **LSV, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France**
goubault@lsv.ens-cachan.fr

Abstract

The homotopy hypothesis was originally stated by Grothendieck [13]: topological spaces should be "equivalent" to (weak) ∞-groupoids, which give algebraic representatives of homotopy types. Much later, several authors developed geometrizations of computational models, e.g. for rewriting, distributed systems, (homotopy) type theory etc. But an essential feature in the work set up in concurrency theory, is that time should be considered irreversible, giving rise to the field of directed algebraic topology. Following the path proposed by Porter, we state here a directed homotopy hypothesis: Grandis' directed topological spaces should be "equivalent" to a weak form of topologically enriched categories, still very close to $(\infty,1)$-categories. We develop, as in ordinary algebraic topology, a directed homotopy equivalence and a weak equivalence, and show invariance of a form of directed homology.

1998 ACM Subject Classification G. Mathematics of Computing, F.1.2 [Modes of Computation] Parallelism and concurrency

Keywords and phrases directed algebraic topology, partially enriched categories, homotopy hypothesis, geometric models for concurrency, higher category theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.9

1 Introduction

A central motivation in algebraic topology is to be able, through algebraic calculations, to classify topological spaces, up to homeomorphisms, or up to some shape equivalence (homotopy equivalence, or equivalence under "elastic deformations"). Similarly, computer science abounds with notions of equivalences, first and foremost, bisimulation equivalences between concurrent processes [20]. It has been observed that some of these equivalences are very geometric in nature: Pratt [23] and van Glabbeek [27], followed by many, among which [9, 12], advocated for cubical models and topological models of the execution space of concurrent systems.

Topological spaces naturally give rise to a higher dimensional category: paths, homotopies, higher homotopies provide the cells of such a structure. Moreover, by nature those cells are invertible up to higher cells: for example, paths are invertible up to homotopy, etc. Hence it is natural to ask the following question, which has become known as the homotopy hypothesis question [13]: "to what extent is the structure of spaces mirrored by that of ∞-groupoids ?"

© Jérémy Dubut, Éric Goubault, and Jean Goubault-Larrecq;
licensed under Creative Commons License CC-BY

A modern answer to this question uses the language of Quillen's model structure: weak homotopy types are modeled by structures that can be interpreted as being ∞-groupoids.

The equivalences of interest in computer science are *directed*, instead: there is a direction of time, and it must be preserved by deformation. Porter [21, 22] proposed to look at a directed analogue of the homotopy hypothesis. Directed spaces also give a natural structure of higher categories (dipaths, dihomotopies, higher dihomotopies), but dipaths are not meant to be invertible up to dihomotopy. The directed homotopy hypothesis should correspond to some question of the form **"To what extent are directed spaces the same as $(\infty, 1)$-categories?"** and should have an answer in the language of model categories as well, by comparing directed spaces and simplicial categories, a model of $(\infty, 1)$-categories.

Our goal is to give an answer to that question. We review Porter's approach in Section 3, and show some of its limitations. We then design a proposal of directed homotopy equivalence based on directed deformation retracts along inessential dipaths (Section 4). Finally, we reformulate the directed homotopy hypothesis, fixing the limitations of Porter' proposal, by designing weak equivalences for a weak version of enriched categories, based on partial enrichment and directed components, for which we prove adequacy with respect to dihomotopy equivalence (Section 5) – our main result.

2 A lexicon of equivalences

There will be many notions of equivalences in this paper and it will be important that they are not mixed up, so here is a brief lexicon:

- **weak equivalence:** the generic name for an element of the distinguished class of morphisms of a model category [24] which are meant to be turned into isomorphisms by the categorical process of localization.
- **homotopy equivalence:** invertible continuous function up to homotopy. They are the weak equivalences in the Strøm model structure on topological spaces [26].
- **weak homotopy equivalence:** continuous function that induces isomorphisms of homotopy groups in every dimensions [14]. A homotopy equivalence is a weak homotopy equivalence. They are the weak equivalences in the Quillen-Serre model structure on topological spaces. All those notions are *un*directed.
- **naive dihomotopy equivalence:** invertible dmaps (see Section 3) up to dihomotopy.
- **dihomotopy equivalence:** to be defined in Section 4.2. They will be our generalization of homotopy equivalence in the directed case, and should not be confused with other notions of (almost) the same name (e.g. [25, 12]).
- **weak equivalence of partially enriched categories:** to be defined in Section 5.3. They will be a modification of weak equivalences of enriched categories [2].
- **weak dihomotopy equivalence:** to be defined in Section 5.3. They will be our modification of Porter's proposal (see Section 2), based on weak equivalences of partially enriched categories.
- **strong equivalence of partially enriched categories:** to be defined in Section 5.4. They will be a generalization of equivalence of categories (but not of equivalence of enriched categories).

3 From homotopy hypothesis to Porter's directed homotopy hypothesis

Topological spaces naturally yield a structure of ∞-groupoids i.e. a structure with 0-cells or objects, 1-cells or morphisms between 0-cells, 2-cells or morphisms between 1-cells, and

so on, such that every n-cell is invertible up to $n+1$-cells for $n \geq 1$. From a topological space X, we can construct an ∞-groupoid by taking as 0-cells, the points of X, 1-cells are the paths, i.e., continuous functions from the unit segment $I = [0,1]$ to X, 2-cells are the homotopies between paths, i.e. continuous functions $H : I \times I \longrightarrow X$ such that $H(_, 0)$ and $H(_, 1)$ are constant maps, ..., higher cells are the higher homotopies. This is a ∞-groupoid since n-homotopies are invertible up to $n+1$-homotopies for all n. For example, a path γ is invertible up to homotopy since, if we note γ^{-1} the path $t \mapsto \gamma(1-t)$ and $\star$ the concatenation of paths, $\gamma \star \gamma^{-1}$ and $\gamma^{-1} \star \gamma$ are homotopic to constant paths, i.e., there is a homotopy H such that $H(0, _) = \gamma \star \gamma^{-1}$ and $H(1, _)$ is constant (equal to $\gamma(0)$), and similarly for $\gamma^{-1} \star \gamma$.

There are many ways to model (in the sense of model categories) ∞-groupoids. One of the simplest ones is Kan complexes, i.e. simplicial sets K for which "every horn has a filler". A horn is simply a simplicial map from $\Delta_i[n]$, the union of the faces of the standard n-simplex $\Delta[n]$, except the i-th one, to K. Having a filler means that this map extends to a simplicial map from $\Delta[n]$ to K. In the language of model categories, they are precisely the fibrant objects of the Kan-Quillen model structure on simplicial sets.

The singular simplicial complex functor provides a Kan complex from a topological space X, where the n-cells are the continuous maps from the geometric standard n-simplex to X. This functor has a left adjoint, the geometric realization that build a topological space from a simplicial set by glueing simplices together. The homotopy hypothesis can then be formulated as follows [24]: this adjunction is a Quillen-equivalence between the Quillen-Serre model structure on topological spaces (whose weak equivalences are the weak homotopy equivalences) and the Kan-Quillen model structure on simplicial sets (whose weak equivalences are simplicial maps that induce weak homotopy equivalences on the geometric realization). This formulation has many consequences: first, weak homotopy types are modeled by ∞-groupoids; secondly, one can compare topological spaces up to weak homotopy equivalence by comparing Kan complexes.

Based on this, Porter [21, 22] proposed a directed homotopy hypothesis for directed spaces. Let us first recall a few basic notions from directed topology (as in e.g. [12, 9]). A **directed space** (or **dspace** for short), is a topological space X, together with a subset of paths P_X, called the **directed paths** (or **dipaths**), satisfying the following:

- every constant path is in P_X;
- P_X is closed under concatenation;
- P_X is closed under non-decreasing reparametrization, i.e., if $\gamma \in P_X$ and $r : I \longrightarrow I$ is a continuous non-decreasing function, then $\gamma \circ r \in P_X$.

A **dmap** $f : X \longrightarrow Y$ is a continuous function such that for every $\gamma \in P_X$, $f \circ \gamma \in P_Y$. We note *dTop*, the category of dspaces and dmaps. A **dihomotopy of dipaths of** X is a homotopy between paths $H : I \times I \longrightarrow X$ such that for every $t \in I$, $H(t, _)$ is a dipath. More generally, one can define n-dihomotopies. Contrary to topological spaces, dipaths need not be invertible up to dihomotopy: define $\overrightarrow{I}$ as the dspace I whose dipaths are the non-decreasing paths. The identity function of I is a dipath going from 0 to 1, but there is no dipath from 1 to 0, and so it cannot have an inverse modulo homotopy. Hence if a topological space is to be thought of as being "the same as" an ∞-groupoid, then a dspace, whose dipaths are not invertible up to dihomotopies, should be the same as an $(\infty, 1)$-category, namely, a ∞-groupoid whose 1-cells are not required to be invertible up to 2-cells.

Much as ∞-groupoids, there are many ways to model $(\infty, 1)$-categories. Two are really close to Kan complexes: quasi-categories [16] (weak Kan complexes in the sense that only "inner horns", i.e. i-horns for $i \neq 0$ and $i \neq n$ are required to have fillers) and enriched

categories over Kan complexes [2]. Porter proposed to follow Quillen's program, by using the latter. Given a dspace, one can construct [22] the following simplicial category $\mathbb{T}(X)$ (which is actually enriched in Kan complexes), called the **trace category**:

- its objects are the points of X;
- for every pair of points (x, y), the simplicial set $\mathbb{T}(X)(x, y)$ is the singular simplicial complex of the trace space $\overrightarrow{T}(X)(x, y)$, which is the space of dipaths up to non-decreasing reparametrizations with the quotient topology of the compact-open topology [8].

One can then compare dspaces through the lens of the model structure of simplicial categories [22]. The weak equivalences are those simplicially enriched functors $F : C \longrightarrow D$ such that:

- for every pair of objects (c, c') of C, the simplicial map $F_{c,c'} : C(c, c') \longrightarrow D(F(c), F(c'))$ induces a weak homotopy equivalence between geometric realizations (i.e., is a weak equivalence in the Kan-Quillen model structure);
- F induces an equivalence of categories $\pi_0(F) : \pi_0(C) \longrightarrow \pi_0(D)$ where $\pi_0(C)$, the category of components of C, is obtained from C by replacing $C(c, c')$ by the set of 0-cells of $C(c, c')$ quotient by the 1-cells (or equivalently, the set of path-connected components of the geometric realization of $C(c, c')$).

Even if this program seems natural and foreshadows the existence of a model structure for dspaces (which is a very enticing perspective), this method has its own limitations. Let us consider again the directed segment $\overrightarrow{I}$. In every obvious (weak) directed homotopy equivalence (see also later), $\overrightarrow{I}$ should be equivalent to a point space $*$. So, we expect that they will have the same trace category up to weak equivalence, which is not the case:

- $\mathbb{T}(\overrightarrow{I})(1, 0)$ is empty while $\mathbb{T}(*)(*, *)$ is not and so cannot be weakly equivalent (in the Kan-Quillen model structure);
- $\pi_0(\mathbb{T}(\overrightarrow{I}))$ is isomorphic to the poset $(I, \leq)$ and so cannot be equivalent to $\pi_0(\mathbb{T}(*))$.

We must therefore better understand the meaning of (weak) directed homotopy type to make the whole dihomotopy hypothesis programme work fine.

4 Directed homotopy equivalences

4.1 A naive directed homotopy equivalence

There are numerous proposals for directed homotopy equivalences [12, 25, 18, 10, 5], but none has yet gained unequivocal approval in the community. The simplest is the textual generalization of the classical definition of homotopy equivalence in algebraic topology. A **homotopy** is a continuous function $H : I \times X \longrightarrow Y$. We then say that two continuous functions $f, g : X \longrightarrow Y$ are **homotopic** [14], if there is a homotopy H such that $H(0, _) = f$ and $H(1, _) = g$. This is an equivalence relation, compatible with composition. We then write $HoTop$ for the category of topological spaces and homotopy classes of continuous functions. We call **homotopy equivalence** a continuous function $f : X \longrightarrow Y$ whose homotopy class is an isomorphism in $HoTop$, i.e., such that there is a continuous function $g : Y \longrightarrow X$ such that $f \circ g$ and $g \circ f$ are homotopic to identities. We say that two spaces are **homotopy equivalent** if there is a homotopy equivalence between them.

Similarly, a ***di*homotopy** [9] is a continuous function $H : I \times X \longrightarrow Y$, *such that for every* $t \in I$, $H(t, _)$ *is a dmap.* We say that two *dmaps* $f, g : X \longrightarrow Y$ are ***di*homotopic** if there is a *di*homotopy H such that $H(0, _) = f$ and $H(1, _) = g$. This is an equivalence relation, compatible with composition. We call ***naive di*homotopy equivalence** a *dmap* $f : X \longrightarrow Y$ which is invertible up to *di*homotopy, i.e., such that there is a *dmap* $g : Y \longrightarrow X$ such that $f \circ g$ and $g \circ f$ are *di*homotopic to identities. We say that two *d*spaces are ***naive di*homotopy equivalent** if there is a *naive di*homotopy equivalence between them.

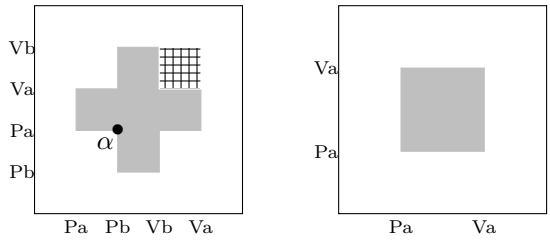


Figure 1 dspaces SF and HS.

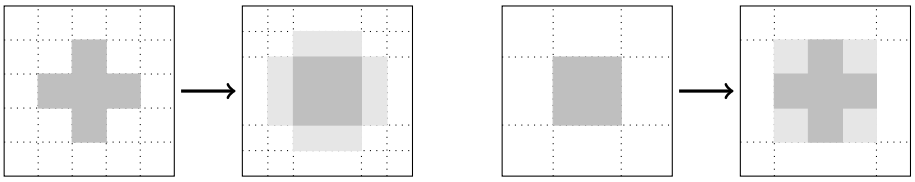


Figure 2 dmap from SF to HS (left) and from HS to SF (right).

▶ **Example 1.**

1. The two dspaces in Figure 1 come from the geometric semantics of programs with semaphores [9], one process is taking locks on shared objects a and b (P actions) before relinquishing them (V actions), the second process is doing the same, in reverse order on objects.

   Those dspaces are subspaces of $\mathbb{R}^2$ whose points are within the white part in the square (the grey part represents the forbidden states of the program) and whose dipaths are non decreasing paths for the componentwise ordering on $\mathbb{R}^2$. They are naive dihomotopy equivalent since there are two maps, depicted in Figure 2, that form a naive dihomotopy equivalence.

   The points in light grey are the points which do not belong to the image of those maps. The problem is that those two programs are quite different: SF has a dead-lock in α and inaccessible states (depicted in Figure 1, as the hatched upper right concavity), while HC does not. Topologically, they do not have the same (directed) components in the sense of [11].

2. Next, let us consider the dspace in Figure 3, which we call the Fahrenberg matchbox [7]. Geometrically, this is an empty cube without bottom face. Its dipaths are the paths that only go from bottom to top and from front to back. We expect it to be non-dihomotopically equivalent to a point because it has a non-trivial dihomotopy type (although it is contractible in the usual sense). Indeed, consider the two dipaths depicted in Figure 3. They are not dihomotopic because the only way to deform continuously one into the other is to go through the upper face, and one of the intermediate paths (namely, any such path that goes through the topmost face) will fail to be a dipath.

   However, M is naive dihomotopy equivalent to its upper face (so to a point), a dihomotopy is depicted in Figure 4.

   More precisely, the dmap f, which maps any point of M to the point of T just above of it, is a naive dihomotopy equivalence, whose inverse modulo dihomotopy is the embedding g of T into M. Hence, $f \circ g = id_T$ and a dihomotopy from id_M to $g \circ f$ is depicted in Figure 4.

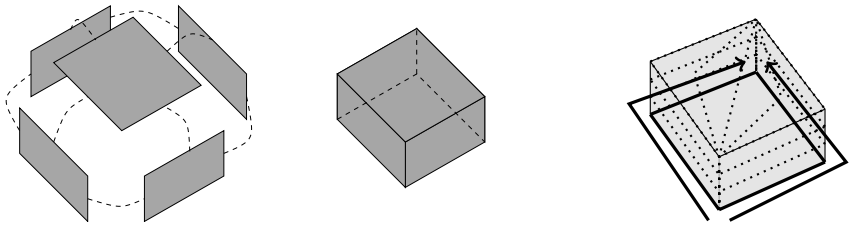


Figure 3 Fahrenberg's matchbox M and two non-dihomotopic dipaths.

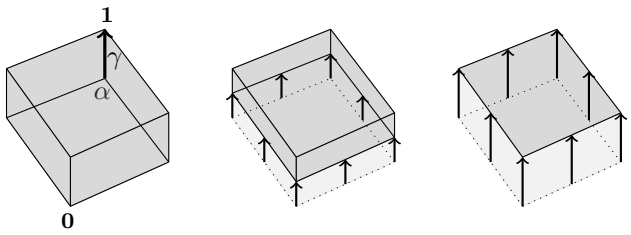


Figure 4 Naive equivalence between the Fahrenberg's matchbox M and its upper face T.

4.2 Deformation retracts and dihomotopy equivalence

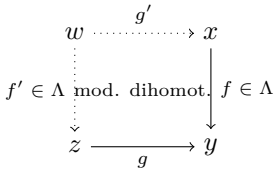
There is another way to define homotopy equivalence in classical algebraic topology, which will prove better for getting the right definition in directed algebraic topology. A homotopy equivalence can be formalized by the notion of deformation retract [14]. Let X be a space and A be a subspace of X. We say that A is a **deformation retract** of X if and only if there is a homotopy $H : I \times X \longrightarrow X$ such that $H(0, _) = id_X$, for every $t \in I$ and every $a \in A$, $H(t, a) = a$ and for all $x \in X$, $H(1, x) \in A$. This definition of deformation retract states that the embedding of A into X is a homotopy equivalence with inverse modulo homotopy $H(1, _)$. In fact, deformation retracts characterize homotopy equivalence in the following sense [14]: two spaces X and Y are homotopy equivalent iff there is a space Z such that X and Y are deformation retracts of Z.

A homotopy being the same as a continuous function from X to $Top(I, Y)$, where $Top(I, Y)$ is the set of paths in Y equipped with the compact-open topology, one can define "directed" deformation retracts as continuous functions from X to P_X, (equipped with the subspace topology) satisfying the same kind of axioms as deformation retracts satisfy. But one must be careful: the dihomotopy depicted in Figure 4 will be a directed deformation retract in this sense. The main problem is that the dipaths along which we deform (i.e., the dipaths in the image of the deformation retracts) will not preserve the fact that two dipaths are not dihomotopic (for example, dipath γ), and more generally the (classical) homotopy type of space of dipaths, while it is the case in the non-directed setting. Hence, some form of "components" as in [11] should underly the definition of directed deformation retract. As a side effect, we will also naturally get to define two notions of deformation retracts, one in the future, one in the past.

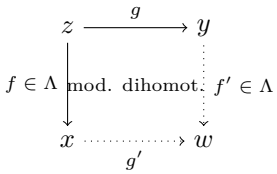
In the following, we write $\overrightarrow{\mathbb{P}}(X)(x, y)$ for the set of **dipaths of** X **from** x **to** y, namely dipaths γ of X such that $\gamma(0) = x$ and $\gamma(1) = y$, equipped with the compact-open topology. Imitating [11], we call a **Yoneda system of dipaths of** X any subset Λ of P_X such that:

- Λ is closed under concatenation and dihomotopy;
- for every $\gamma : x \longrightarrow y \in \Lambda$, for every $z \in X$ such that $\overrightarrow{\mathbb{P}}(X)(y, z) \neq \varnothing$, the function $\gamma \star _ : \overrightarrow{\mathbb{P}}(X)(y, z) \longrightarrow \overrightarrow{\mathbb{P}}(X)(x, z)$, $\delta \longmapsto \gamma \star \delta$ is a homotopy equivalence;

- for every $\gamma : x \longrightarrow y \in \Lambda$, for every $w \in X$ such that $\overrightarrow{\mathbb{P}}(X)(w,x) \neq \varnothing$, the function $_ \star \gamma : \overrightarrow{\mathbb{P}}(X)(w,x) \longrightarrow \overrightarrow{\mathbb{P}}(X)(w,y)$, $\delta \longmapsto \delta \star \gamma$ is a homotopy equivalence;
- Λ has the right Ore condition modulo dihomotopy, i.e., for every $f : x \longrightarrow y \in \Lambda$ and every dipath $g : z \longrightarrow y$ in X there are $f' : w \longrightarrow z \in \Lambda$ and a dipath $g' : w \longrightarrow x$ in X for some w such that $g' \star f$ and $f' \star g$ are dihomotopic;



- Λ has the left Ore condition modulo dihomotopy, i.e., for every $f : x \longrightarrow y \in \Lambda$ and every dipath $g : x \longrightarrow z$ in X there are $f' : z \longrightarrow w \in \Lambda$ and a dipath $g' : x \longrightarrow w$ in X for some w such that $g \star f'$ and $f \star g'$ are dihomotopic.



▶ **Lemma 2.** *The set of Yoneda systems of dipaths of X is a complete lattice for inclusion. We note $\mathfrak{I}(X)$ the largest such system and call its elements* ***inessential dipaths****.*

Let X be a dspace and A be a sub-dspace of X, i.e., a sub-topological space $A \subseteq X$ whose dipaths are the dipaths of X with image in A. We say that A is a **future deformation retract** of X if there is a continuous function $H : X \longrightarrow \mathfrak{I}(X)$ ($\mathfrak{I}(X)$ is equipped with the subspace topology of $\mathbf{Top}(I, X)$) such that:

- for every $x \in X$, $H(x)(0) = x$;
- for every $a \in A$ and $t \in I$, $H(a)(t) = a$;
- for every $x \in X$, $H(x)(1) \in A$;
- for every $t \in I$, the map $H_t : X \longrightarrow X$, $x \longmapsto H(x)(t)$ is a dmap;
- for every dipath δ of A from z to $H_1(x)$ there is a dipath γ of X from y to x with $H_1(y) = z$ and $H_1 \circ \gamma$ and δ are dihomotopic.

We stress here the fact that H must be with values in the inessential dipaths $\mathfrak{I}(X)$. Similarly, we define **past deformation retracts** by switching the role of 1 and 0 in the previous definition. We then say that two dspaces are **directed homotopy equivalent** if there is a zigzag of future and past deformation retracts between them.

▶ **Example 3.**

1. Observe that past deformation retracts (resp. future deformation retracts) between topological spaces (i.e. dpsaces whose set of dipaths contains all paths) coincide with non-directed deformation retracts. In particular, if two topological spaces are homotopically equivalent then they are dihomotopically equivalent. The converse also holds.
2. $\{1\}$ is a future deformation retract of $\overrightarrow{I}$. Indeed, the function $H : \overrightarrow{I} \longrightarrow \mathfrak{I}(\overrightarrow{I})$, $s \longmapsto (t \longmapsto (1-t)s + t)$ satisfies the conditions above. Similarly, $\{0\}$ is a past deformation retract of $\overrightarrow{I}$. More generally, every past face $\overrightarrow{I}^k \times \{0\} \times \overrightarrow{I}^l$ (resp. future face $\overrightarrow{I}^k \times \{1\} \times \overrightarrow{I}^l$) is a past (resp. future) deformation retract of the directed cube $\overrightarrow{I}^{k+l+1}$.

3. Is the deformation depicted in Figure 4 a future deformation retract from M to its upper face T ? The answer is no because it is not with values in $\mathfrak{I}(M)$. Indeed, the dipath γ from α to $\mathbf{1}$ does not induce a homotopy equivalence between spaces of dipaths. The space $\overrightarrow{\mathbb{P}}(M)(\mathbf{0},\alpha)$ of dipaths from $\mathbf{0}$ to α is homotopically equivalent to a two point space, because there are two such dipaths that are not dihomotopic, while the space $\overrightarrow{\mathbb{P}}(M)(\mathbf{0},\mathbf{1})$ of dipaths from $\mathbf{0}$ to $\mathbf{1}$ is contractible, so they cannot be homotopically equivalent.

4.3 A homological invariant of dihomotopy equivalence: natural homology

Let us recall the notion of natural homology of [6, 25]. Given a dspace X we form as previously the fundamental category $\overrightarrow{\pi_1}(X)$ whose objects are points of X and whose morphisms from x to y are classes modulo dihomotopy of dipaths from x to y. Then, we take its **category of factorizations**, noted $\mathcal{F}_X$ whose objects are classes modulo dihomotopy of dipaths and whose morphisms from the class $[\gamma]$ with γ a dipath from x to y to the class $[\gamma']$ with γ' a dipath from x' to y' are pairs of classes $([\alpha],[\beta])$ with α from x' to x and β from y to y' such that $[\alpha \star \gamma \star \beta] = [\gamma']$. Composition is concatenation and identities are pairs of classes of constant dipaths. We then define the **natural dipath functor of** X, the functor $\overrightarrow{\mathcal{P}}(X) : \mathcal{F}_X \longrightarrow HoTop$ which maps:

- every class $[\gamma]$ with γ from x to y to $\overrightarrow{\mathbb{P}}(X)(x,y)$;
- every extension $([\alpha],[\beta])$ to the class modulo homotopy of the map $\delta \longmapsto (\alpha \star \delta) \star \beta$.

We can then form the **natural homology of** X by composing with the singular homology functor. The definition of [6] is based on taking the category of factorizations of the trace category, instead of the fundamental category. This gives a "bisimilar" notion, when X is a pospace (the setting of [6]) and will be more convenient to work with here. This notion of bisimilarity of diagrams with values in Abelian groups (or more generally in any fixed category) is fully defined in [6], based on the framework of open maps [17], and is designed for comparing directed homology of pospaces. It goes as follows. The context is that of **small diagrams with values in a category** $\mathcal{M}$ which are functors from any small category $\mathcal{C}$ to the category $\mathcal{M}$. A **morphism of diagrams** from $F : \mathcal{C} \longrightarrow \mathcal{M}$ to $G : \mathcal{D} \longrightarrow \mathcal{M}$ is a pair (Φ,σ) of a functor $\Phi : \mathcal{C} \longrightarrow \mathcal{D}$ and a natural transformation $\sigma : F \longrightarrow G \circ \Phi$. We note **Diag($\mathcal{M}$)** the category of small diagrams with values in Abelian groups and morphisms of diagrams.

A morphism of diagrams (Φ,σ) from $F : \mathcal{C} \longrightarrow \mathcal{M}$ to $G : \mathcal{D} \longrightarrow \mathcal{M}$ is an **open map** [6] if and only if:

- σ is a natural isomorphism;
- Φ is surjective-on-objects;
- for every morphism $j : F(c) \longrightarrow d$ of $\mathcal{D}$ there is a morphism $i : c \longrightarrow c'$ of $\mathcal{C}$ such that $F(i) = j$.

Two diagrams $F : \mathcal{C} \longrightarrow \mathcal{M}$ and $G : \mathcal{D} \longrightarrow \mathcal{M}$ are **bisimilar** if there is span of open maps between them, i.e., there are a diagram $H : \mathcal{E} \longrightarrow \mathcal{M}$ and two open maps $(\Phi,\sigma) : H \longrightarrow F$ and $(\Psi,\tau) : H \longrightarrow G$.

We can then prove that natural dipath functors are invariant modulo dihomotopy equivalence when we compare them up to bisimilarity:

▶ **Theorem 4.** *If two dspaces are dihomotopically equivalent then their natural dipath functors (and so their natural homology) are bisimilar (in* ***Diag(****HoTop****)****).*

Since the Fahrenberg matchbox and a point have non-bisimilar natural homology [6], they cannot be dihomotopically equivalent.

5 Weak directed homotopy equivalence

In Section 3, we have seen the limitation of too simple an implementation of Porter's programme on the example of the directed segment: empty path spaces are not well handled because we are requiring a weak homotopy equivalence for each pair of points and because the (ordinary) component category is somehow too rigid. We fix those two problems in this section. First, in Section 5.1, we introduce a notion of partially enriched categories, i.e., enriched categories where only some morphism objects between two objects are defined (intuitively, the non-empty ones). Secondly, we replace components by directed components in the style of [11]. Altogether, this defines a weak dihomotopy equivalence which is an invariant of dihomotopy equivalence (see Section 5.3).

5.1 Partially enriched categories and the dipath category

In the following, $\mathcal{V}$ is a monoidal category with $\otimes$ as tensor product, U as unit, $\alpha_{A,B,C} : (A\otimes B)\otimes C \longrightarrow A\otimes(B\otimes C)$ as associator, $\lambda_A : U\otimes A \longrightarrow A$ as left unit and $\rho_A : A\otimes U \longrightarrow A$ as right unit. We will mainly consider the case of Top (the category of topological spaces and continuous functions), $HoTop$ (category of topological spaces and continuous functions modulo homotopy) and Ab (category of Abelian groups and morphisms of groups) with their Cartesian structure.

A (small $\mathcal{V}$-)**partially enriched category** $\mathcal{C}$ consists of the following data:

- a set $Ob(\mathcal{C})$ of objects;
- a preorder $\leq$ on $Ob(\mathcal{C})$ called **domain**;
- for every pair $c \leq c'$ of objects of $\mathcal{C}$, an object $\mathcal{C}(c,c')$ of $\mathcal{V}$;
- for every triple $c \leq c' \leq c''$ of objects of $\mathcal{C}$, a composition morphism in $\mathcal{V}$ $\circ_{c,c',c''} : \mathcal{C}(c,c')\otimes\mathcal{C}(c',c'') \longrightarrow \mathcal{C}(c,c'')$;
- for every object c of $\mathcal{C}$, a unit morphism in $\mathcal{V}$ $u_c : U \longrightarrow \mathcal{C}(c,c)$.

satisfying:

- (associativity): for every quadruple $c \leq c' \leq c'' \leq c'''$ of objects of $\mathcal{C}$, the following diagram commutes:

$$\begin{array}{ccc}
(\mathcal{C}(c,c')\otimes\mathcal{C}(c',c''))\otimes\mathcal{C}(c'',c''') & \xrightarrow{\circ_{c,c',c''}\otimes id} & \mathcal{C}(c,c'')\otimes\mathcal{C}(c'',c''') \\
\downarrow{\scriptstyle \alpha_{\mathcal{C}(c,c'),\mathcal{C}(c',c''),\mathcal{C}(c,c'')}} & & \\
\mathcal{C}(c,c')\otimes(\mathcal{C}(c',c'')\otimes\mathcal{C}(c'',c''')) & & \downarrow{\scriptstyle \circ_{c,c'',c'''}} \\
\downarrow{\scriptstyle id\otimes\circ_{c',c'',c'''}} & & \\
\mathcal{C}(c,c')\otimes\mathcal{C}(c',c''') & \xrightarrow{\circ_{c,c',c'''}} & \mathcal{C}(c,c''')
\end{array}$$

- (unit): for every pair $c \leq c'$ of objects of $\mathcal{C}$, the following diagrams commute:

$$\begin{array}{ccccc}
U\otimes\mathcal{C}(c,c') & \xrightarrow{\lambda_{\mathcal{C}(c,c')}} & \mathcal{C}(c,c') & \xleftarrow{\rho_{\mathcal{C}(c,c')}} & \mathcal{C}(c,c')\otimes U \\
\downarrow{\scriptstyle u_c\otimes id} & & \downarrow{\scriptstyle id} & & \downarrow{\scriptstyle id\otimes u_{c'}} \\
\mathcal{C}(c,c)\otimes\mathcal{C}(c,c') & \xrightarrow{\circ_{c,c,c'}} & \mathcal{C}(c,c') & \xleftarrow{\circ_{c,c',c'}} & \mathcal{C}(c,c')\otimes\mathcal{C}(c',c')
\end{array}$$

The axioms are the same as for $\mathcal{V}$-enriched categories [4], except for the fundamental role played by the domain $\leq$ in every clause. Trivially, an enriched category is a partially enriched category whose domain is $Ob(\mathcal{C})\times Ob(\mathcal{C})$. One should note that partially enriched categories

in Top, in $HoTop$ and in $Simp$ (category of simplicial sets and simplicial maps) are still very close to $(\infty, 1)$-categories but also to Gaucher's flows [10], which were introduced for similar motivations.

Our main use of partially enriched categories will be the following. Given a dspace (X, P_X), we construct a partially enriched category over $HoTop$ called the **dipath category** and written $\overrightarrow{\mathbb{P}}(X)$:

- its objects are points of X;
- its domain, called the **accessibility preordering** is $x \leq y$ iff there a dipath from x to y;
- for $x \leq y$, $\overrightarrow{\mathbb{P}}(X)(x, y)$ is the set of dipaths from x to y equipped with the compact-open topology, as already defined in Section 4.2;
- composition is the class modulo homotopy of the concatenation $(\gamma, \delta) \mapsto \gamma \star \delta$;
- for $x \in X$, the unit morphism u_x is the class modulo homotopy of the continuous function $\{*\} \longrightarrow \overrightarrow{\mathbb{P}}(X)(x, x)$, $* \longmapsto c_x$.

A **partially enriched functor** $F : \mathcal{C} \longrightarrow \mathcal{D}$ between partially enriched categories is the following data:

- a monotonic function $F : Ob(\mathcal{C}) \longrightarrow Ob(\mathcal{D})$;
- for every pair $c \leq c'$ of objects, a morphism $F_{c,c'} : \mathcal{C}(c, c') \longrightarrow \mathcal{D}(F(c), F(c'))$ in $\mathcal{V}$;

satisfying that:

- for every triple $c \leq c' \leq c''$ of objects of $\mathcal{C}$, the following diagram commutes:

$$\begin{array}{ccc} \mathcal{C}(c,c') \otimes \mathcal{C}(c',c'') & \xrightarrow{\circ_{c,c',c''}} & \mathcal{C}(c,c'') \\ \downarrow{\scriptstyle F_{c,c'} \otimes F_{c',c''}} & & \downarrow{\scriptstyle F_{c,c''}} \\ \mathcal{D}(F(c),F(c')) \otimes \mathcal{D}(F(c'),F(c'')) & \xrightarrow{\circ_{F(c),F(c'),F(c'')}} & \mathcal{D}(F(c),F(c'')) \end{array}$$

- for every object c of $\mathcal{C}$, the following diagram commutes:

$$\begin{array}{ccc} U & \xrightarrow{u_c} & \mathcal{C}(c,c) \\ & \searrow^{u_{F(c)}} & \downarrow{\scriptstyle F_{c,c}} \\ & & \mathcal{D}(F(c),F(c)) \end{array}$$

The partially enriched functors between enriched categories seen as partially enriched categories are exactly the enriched functors. We note **PeCat($\mathcal{V}$)** the category of small $\mathcal{V}$-partially enriched categories and partially enriched functors. Note that $\overrightarrow{\mathbb{P}}$ extends to a functor from $dTop$ to **PeCat(**$HoTop$**)**.

5.2 Category of components

We recover the **fundamental category** $\overrightarrow{\pi_1}(X)$ of a dspace X [12] by applying the ordinary path-connected components functor π_0 to its category of dipaths $\overrightarrow{\mathbb{P}}(X)$:

- its objects are points of X;
- the morphisms from x to y are the elements of $\pi_0(\overrightarrow{\mathbb{P}}(X)(x, y))$, i.e., the path-connected components of $\overrightarrow{\mathbb{P}}(X)(x, y)$ if $x \leq y$ and $\varnothing$ otherwise;
- the composition is the function: $\circ : \pi_0(\overrightarrow{\mathbb{P}}(X)(y, z)) \times \pi_0(\overrightarrow{\mathbb{P}}(X)(x, y)) \longrightarrow \pi_0(\overrightarrow{\mathbb{P}}(X)(x, z))$ such that $[f] \circ [g] = [g \star f]$;
- the identity of x is the path-connected components of the constant dipath, i.e., $[c_x]$.

Morphisms of $\overrightarrow{\pi_1}(X)$ are exactly dipaths modulo dihomotopy. More generally, we can define similarly the **fundamental category $\overrightarrow{\pi_1}(\mathcal{C})$ of a partially enriched category $\mathcal{C}$ on** $HoTop$ which extends to a functor from **PeCat(**$HoTop$**)** to Cat.

From this fundamental category, we can define a category of components as in [11]. The idea is to define a class of inessential morphisms, which is the largest class of morphisms whose compositions to the left and to the right are bijections, that has a left and right calculi of fractions (exactly like in the definition of inessential dipaths in Section 4.2). Then those morphisms can be inverted to give the **category of components $\overrightarrow{\pi_0}(\mathcal{C})$ of a category** $\mathcal{C}$ and, by extension, of a partially enriched category $\mathcal{C}$ on **HoTop** (resp. a dspace X) by $\overrightarrow{\pi_0}(\mathcal{C}) = \overrightarrow{\pi_0}(\overrightarrow{\pi_1}(\mathcal{C}))$ (resp. $\overrightarrow{\pi_0}(X) = \overrightarrow{\pi_0}(\overrightarrow{\pi_1}(X))$).

Explicitly, given a small category $\mathcal{C}$, we define a **Yoneda system Λ of morphisms of** $\mathcal{C}$ as a subset of morphisms of C such that:

- Λ is closed under composition;
- for every $f : c \longrightarrow c' \in \Lambda$, for every object c'' of $\mathcal{C}$ such that $\mathcal{C}(c', c'') \neq \varnothing$, the function $_ \circ f : \mathcal{C}(c', c'') \longrightarrow \mathcal{C}(c, c'')$ $g \longmapsto g \circ f$ is a bijection;
- for every $f : c \longrightarrow c' \in \Lambda$, for every object c'' of $\mathcal{C}$ such that $\mathcal{C}(c'', c) \neq \varnothing$, the function $f \circ _ : \mathcal{C}(c'', c) \longrightarrow \mathcal{C}(c'', c')$ $g \longmapsto f \circ g$ is a bijection;
- Λ has the right Ore condition, i.e., for every $f : x \longrightarrow y \in \Lambda$ and every $g : z \longrightarrow y \in \mathcal{C}$ there are $f' : w \longrightarrow z \in \Lambda$ and $g' : w \longrightarrow x \in \mathcal{C}$ for some w such that $f \circ g' = g \circ f'$
- Λ has the left Ore condition, i.e., for every $f : x \longrightarrow y \in \Lambda$ and every $g : x \longrightarrow z \in \mathcal{C}$ there are $f' : z \longrightarrow w \in \Lambda$ and $g' : x \longrightarrow w \in \mathcal{C}$ for some w such that $f' \circ g = g' \circ f$.

In particular, $(\mathcal{C}, \Lambda)$ has left and right calculi of fractions [3]. This is related to the definition of [11], which additionally requires closure by pullbacks/pushouts in order to obtain a Van-Kampen theorem.

▶ **Lemma 5.** *The set of Yoneda systems of morphisms of $\mathcal{C}$ is a complete lattice for inclusion. We note $\mathfrak{I}(\mathcal{C})$ the largest such system and call its elements **inessential morphisms**. $\mathfrak{I}(\mathcal{C})$ contains all isomorphisms and has the 2-out-3 property, i.e., if two of the three morphisms f, g and $g \circ f$ are in $\mathfrak{I}(\mathcal{C})$, then so is the third. Moreover, $\{[\gamma] \in \overrightarrow{\pi_1}(X) \mid \gamma \in \mathfrak{I}(X)\} \subseteq \mathfrak{I}(\overrightarrow{\pi_1}(X))$.*

Now, we define the **category of components $\overrightarrow{\pi_0}(\mathcal{C})$ of** $\mathcal{C}$, as the localization $\mathcal{C}[\mathfrak{I}(\mathcal{C})^{-1}]$ [3]. One interesting property of this localization is that it is equivalent to the generalized quotient (in the sense of [1]) $\mathcal{C}/\mathfrak{I}(\mathcal{C})$ when $\mathcal{C}$ is loop-free [11]. We stress that this is a remarkable property, for a localization, of being equivalent to a quotient, and this will be particularly useful for examples. In fact, this can be generalized when $\mathfrak{I}(\mathcal{C})$ has a "selection". A **partial selection of a category** $\mathcal{C}$ is a subcategory of $\mathcal{C}$ which is a preorder. A **(total) selection** is a partial selection Σ satisfying moreover that for every pair (c, c') of objects of $\mathcal{C}$, if $\mathcal{C}(c, c')$ is non-empty then so is $\Sigma(c, c')$.

▶ **Theorem 6.** *Let Σ be a total selection of $\mathfrak{I}(\mathcal{C})$. $\overrightarrow{\pi_0}(\mathcal{C})$ is equivalent to the generalized quotient $\mathcal{C}/\Sigma$.*

▶ **Example 7.** We show here a few examples of categories of components of dspaces which are equivalent to quotient of fundamental categories.

1. If X is a pospace (i.e. a topological space equipped with a closed partial-ordering), then the fundamental category $\overrightarrow{\pi_1}(X)$ is loop-free in the sense of [11], and so $\mathfrak{I}(\overrightarrow{\pi_1}(X))$ is itself a selection. We then recover the case of [11]. In particular, SF and HS are pospaces and their category of components are equivalent to the categories depicted in Figure 5 (more precisely, to the categories generated by those graphs with the relation $\circlearrowleft$ representing commutativity). In particular, this shows that SF and HS do not have the same category of components, as claimed in Example 1.

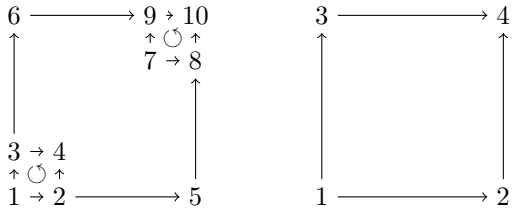


Figure 5 Category of components of SF and HS.

2. Let $S^1 = \{e^{i\theta} \mid \theta \in [0, 2\pi[\}$ be the topological circle. We call non-directed circle the dspace S^1 whose dipaths are all paths. As all paths are invertible modulo homotopy, every morphism of $\overrightarrow{\pi_1}(S^1)$ is an isomorphism and so belongs to $\mathfrak{I}(\overrightarrow{\pi_1}(S^1))$. But, as they are all isomorphisms, $\overrightarrow{\pi_0}(S^1) = \overrightarrow{\pi_1}(S^1)$ which is the fundamental groupoid of the circle. Moreover, $\Sigma = \{[t \mapsto e^{i((1-t)\theta+t\theta')}] \mid \theta, \theta' \in [0, 2\pi[\}$ where $[.]$ is the class modulo homotopy, is a selection of $\mathfrak{I}(S^1)$ and $\overrightarrow{\pi_1}(S^1)/\Sigma$ is the category with one object and $\mathbb{Z}$ as set of morphisms. That category is equivalent to $\overrightarrow{\pi_1}(S^1)$ itself.
3. Let $\overrightarrow{S^1}$ be the dspace whose underlying topological space is S^1 and whose dipaths are paths of the form $t \mapsto e^{i\Phi(t)}$ for some non-decreasing function $\Phi : I \longrightarrow \mathbb{R}$, i.e., paths that only turn anti-clockwise. In this case, the only Yoneda morphisms of $\overrightarrow{S^1}$ are the identities, i.e., dihomotopy class of constant paths. Indeed, they are the only ones that induces bijections between Hom-sets by composition: if you take any non-constant dipath γ, say from $e^{i\theta}$ to $e^{i\theta'}$ then $[\gamma] \circ _ : \overrightarrow{\pi_1}(\overrightarrow{S^1})(e^{i\theta'}, e^{i\theta}) \longrightarrow \overrightarrow{\pi_1}(\overrightarrow{S^1})(e^{i\theta}, e^{i\theta})$ is not surjective since it never reaches the class of the constant path. Hence $\overrightarrow{\pi_0}(\overrightarrow{S^1}) = \overrightarrow{\pi_1}(\overrightarrow{S^1})$.

In this section, we could have defined the category of components in different ways. We have chosen here the classical way, from the fundamental category (just like [2, 11]). But we could have followed the path initiated in the definition of future and past deformation retracts, requiring inessential morphisms to induce homotopy equivalences by composition instead of isomorphisms of path-connected components (cf. conditions 2 and 3). It would have defined a finer notion of components in the sense that, there would have been less inessential morphisms and so less dspaces with the same components. The Theorem 9 would have also hold with this definition, but there would have been some redundancy between this notion of components and the requirement of homotopy equivalences in the definition of weak equivalences.

5.3 Weak equivalences

Imitating [22], we will study dihomotopy types of dspaces using a similar notion of weak equivalences of partially enriched categories. A **weak equivalence between two partially enriched categories $\mathcal{C}$ and $\mathcal{D}$ in** $HoTop$ is a partially enriched functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ which induces an equivalence of categories between $\overrightarrow{\pi_0}(\mathcal{C})$ and $\overrightarrow{\pi_0}(\mathcal{D})$, and such that for every pair $c \leq c'$ in $\mathcal{C}$, $F_{c,c'}$ is an isomorphism, i.e. the homotopy class of a homotopy equivalence. We stress the fact that F induces a functor between the categories of components is not automatic since $\overrightarrow{\pi_0}$ is not a functor. We say that a dmap $f : X \longrightarrow Y$ is a **weak dihomotopy equivalence** if $\overrightarrow{\mathbb{P}}(f)$ is a weak equivalence, and we say that X and Y are **weakly dihomotopy equivalent** if there is a zig-zag of weak equivalences between X and Y.

▶ **Example 8.**

1. As we have said earlier, the directed segment is dihomotopically equivalent to a point and so is weakly equivalent to a point. We have a continuous constant map $c : \overrightarrow{I} \longrightarrow \{1\}$, which is a dmap. Let us prove that this is a weak equivalence. First, for all $x \leq y$, $\overrightarrow{\mathbb{P}}(c)_{x,y}$ is a homotopy equivalence because it is a constant map and $\overrightarrow{\mathbb{P}}(X)(x,y)$ is contractible. Now, it is easy to check that $\mathfrak{I}(\overrightarrow{\pi_1}(\overrightarrow{I})) = \overrightarrow{\pi_1}(\overrightarrow{I})$ and is itself a selection. Therefore $\overrightarrow{\pi_0}(\overrightarrow{I})$ is equivalent to the category with one object and one morphism, namely $\overrightarrow{\pi_0}(\{1\})$. In fact, c induces an equivalence between those two categories.
2. As we have said earlier, the Fahrenberg matchbox M is not dihomotopically equivalent to a point. In fact, they also are non-weakly equivalent. We have seen that there are two dipaths that are not dihomotopic. This implies that $\overrightarrow{\mathbb{P}}(M)(\mathbf{0}, \alpha)$ (with the notation of Figure 4) is homotopically equivalent to a two point spaces. But if two dspaces are weak-equivalent then they have the same homotopy types of non-empty spaces of dipaths. Since the spaces of dipaths of a point are all contractible, the matchbox cannot be weakly equivalent to a point.

▶ **Theorem 9.** *If two dspaces are dihomotopically equivalent then they are weakly equivalent.*

That is, weak dihomotopy equivalence is well-suited to prove that two dspaces are not dihomotopy equivalent (like in the case of T and M). In particular, if two dspaces do not have the same homotopy types of spaces of dipaths then they are not weakly dihomotopy equivalent and therefore not dihomotopy equivalent. The previous examples show in particular that our notion of dihomotopy equivalence is different from the one of [12] (because that one does not distinguish the matchbox from a point) and that the notion of weak equivalence used in [22] is strictly stronger than that introduced above as it distinguishes the directed segment from the point.

5.4 Natural homology and weak equivalences

We have proved that natural homology is an invariant of dihomotopy equivalence (Theorem 4). Is it also an invariant of weak equivalence ? At the time of this article, this is still a conjecture. But as a step in that direction, we observe that bisimilarity is strongly tied to (strong) equivalence of partially enriched categories (Theorem 11 below).

Equivalence of enriched categories is usually defined as in the non-enriched case using (enriched) natural isomorphisms. Using the axiom of choice, this definition is equivalent to the existence of a fully-faithful essentially surjective functor [19]. Nevertheless, we will not use these definitions in the partially enriched case : one problem is that there is no clear non-trivial notion of partially enriched natural transformations. We will rather use the following:

▶ **Lemma 10.** *Two categories are equivalent iff there is a span of fully-faithful surjective-on-objects functors*

We say that a partially enriched functor $F : \mathcal{E} \longrightarrow \mathcal{C}$ is:

- **fully-faithful** if for every pair $e \leq e'$ in $\mathcal{E}$, $F_{e,e'} : \mathcal{E}(e,e') \longrightarrow \mathcal{C}(F(e), F(e'))$ is an isomorphism;
- **surjective** if $F : Ob(\mathcal{E}) \longrightarrow Ob(\mathcal{C})$ is surjective;
- **fibrational** if for every $e \in Ob(\mathcal{E})$ and $c \in Ob(\mathcal{C})$ such that $F(e) \leq c$ there is e' such that $e \leq e'$ and $F(e') = c$.

We call **strong equivalence** any partially enriched functor which is fully-faithful, surjective and fibrational.

We say that two partially enriched categories $\mathcal{C}$ and $\mathcal{D}$ are **strongly equivalent** if there are a partially enriched category $\mathcal{E}$ and a span $F : \mathcal{E} \longrightarrow \mathcal{C}$ and $G : \mathcal{E} \longrightarrow \mathcal{D}$ of strong equivalences.

Without the fibrational condition, this equivalence would be a bit trivial: taking a suitable $\mathcal{E}$ whose domain is equality would make equivalent two partially enriched categories which have the same endomorphisms. Moreover, the strong equivalences between two enriched categories are exactly the fully-faithful surjective enriched functors between them.

Let us look at the case $\mathcal{M} = \mathbf{Ab}$, the category of Abelian groups. In [6] we were only considering diagrams whose domains were pre-orders. Let us call them **po-diagrams** and denote by **PoDiag(Ab)** the full subcategory of those po-diagrams.

Given a diagram $F : \mathcal{C} \longrightarrow \mathbf{Ab}$, define its **unfolding** as the diagram $Unf(F) : Unf(\mathcal{C}) \longrightarrow \mathbf{Ab}$ such that:

- the objects of $Unf(\mathcal{C})$ are non-empty finite sequences $(f_1, ..., f_n)$ of composable morphisms of $\mathcal{C}$, i.e. domain of f_i = codomain of f_{i-1};
- the set of morphisms of $Unf(\mathcal{C})$ from $(f_1, ..., f_n)$ to $(g_1, ..., g_p)$ is $\{(g_{n+1}, ..., g_p)\}$ if $n \leq p$ and for all $i \leq n$, $f_i = g_i$, and is empty otherwise;
- composition of $Unf(\mathcal{C})$ is concatenation;
- identities of $Unf(\mathcal{C})$ are empty sequences;
- $Unf(F)(f_1, ..., f_n) = F(c)$ where c is the codomain of f_n;
- $Unf(F)(g_{n+1}, ..., g_p) = F(g_p \circ \ldots \circ g_{n+1})$.

Given a po-diagram $F : \mathcal{C} \longrightarrow \mathbf{Ab}$, we extend the Grothendieck construction [15] to partially enriched categories in **Ab** as follows:

- the objects are objects of $\mathcal{C}$;
- the domain is $\mathcal{C}$, which we recall is a preorder;
- for $c \leq c'$, $\mathcal{G}(F)(c, c') = F(c')$;
- for $c \leq c' \leq c''$, the composition $\circ_{c,c',c''} : F(c') \times F(c'') \longrightarrow F(c'')$ is the morphism of groups that maps (g', g'') to $g'' + F(c' \leq c'')(g')$, where $c \leq c'$ is the unique morphism from c to c' in $\mathcal{C}$;
- the unit $u_c : \{0\} \longrightarrow F(c)$ is the null morphism.

▶ **Theorem 11.** *Two po-diagrams are bisimilar iff their Grothendieck constructions are equivalent. Two diagrams are bisimilar iff the Grothendieck construction of their unfoldings are equivalent.*

We see Theorem 11 as a tool that we may use later to prove that if the dipath categories of X and Y are weakly equivalent, then their natural homologies are bisimilar.

6 Conclusion and future work

Using partially enriched categories allows us to compare dspaces modulo dihomotopy equivalence. Are partially enriched categories the right models for dspaces modulo dihomotopy equivalence? We hope to have at least conveyed the idea that this should be the case. Are topologically partially enriched categories a nice model of $(\infty, 1)$-categories? Can we conversely understand $(\infty, 1)$-categories using (weak) dihomotopy types of dspaces? Those questions are left to future work.

Also, a recurrent and associated question concerns the algebraic structure of directed homotopy. Ordinary homotopy theories can be described in the framework of Quillen model

categories, giving axioms linking classes of morphisms (in the category of topological spaces, or of simplicial sets, for instance) called weak equivalences, fibrations and cofibrations.

In this paper, we have developed a class of weak equivalences, which could be part of such an axiomatics. More precisely, most of what has been described in Section 5.3 can be parameterized by the class of weak equivalences on the category of topological spaces that is consistent with the standard model category theoretic framework. We used the "stronger" one, Strøm model category where "weak equivalences" are in fact (strong) homotopy equivalences, but indeed, one would be tempted to use weak homotopy equivalences instead. We only "lifted' homotopy equivalences onto the hom-sets of our (partially-enriched) category of dspaces, and of course, we could as well think of lifting fibrations and cofibrations of any closed model category structure on topological spaces, to give a reasonable notion of directed homotopy structure on directed spaces. The resulting classes of morphisms, lifts of weak equivalences, fibrations and cofibrations should verify some of Quillen axioms at least, maybe others. The directed topological community is currently undecided with respect to whether there is a model category of directed spaces which accounts, faithfully, for directed algebraic topological phenomena.

References

1 M. Bednarczyk, A. Borzyszkowski, and W. Pawłowski. Generalized congruences-Epimorphisms in CAT. *TAC*, 5(11):266–280, 1999.
2 J. Bergner. A model category structure on the category of simplicial categories. *Trans. Amer. Math. Soc.*, 2004.
3 F. Borceux. *Handbook of Categorical Algebra 1: Basic Category Theory*. CUP, 1994.
4 F. Borceux. *Handbook of Categorical Algebra 2: Categories and Structures*. CUP, 1994.
5 P. Bubenik and K. Worytkiewicz. A model category for local pospaces. *Homology, Homotopy Appl.*, 8(1):263–292, 2006.
6 J. Dubut, E. Goubault, and J. Goubault-Larrecq. Natural Homology. In *ICALP*, 2015.
7 U. Fahrenberg. Directed homology. *ENTCS*, 100:111–125, 2004.
8 U. Fahrenberg and M. Raussen. Reparametrizations of continuous paths. *JHRS*, 2(2):93–117, 2007.
9 L. Fajstrup, E. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency*. Springer, 2016.
10 P. Gaucher. A model category for the Homotopy Theory of Concurrency. *Homology, Homotopy Appl.*, 5(1):549–593, 2003.
11 E. Goubault and E. Haucourt. Components of the Fundamental Category II. *ACS*, 15(4):387–414, 2007.
12 M. Grandis. *Directed algebraic topology, Models of non-reversible worlds*. CUP, 2009.
13 A. Grothendieck. Pursuing Stacks. Manuscript.
14 A. Hatcher. *Algebraic Topology*. CUP, 2002.
15 P. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*. OUP, 2002.
16 A. Joyal. Notes on quasi-categories. Unpublished.
17 A. Joyal, M. Nielsen, and G. Winske. Bisimulation from open maps. *Inform. Comput.*, 127(2):164–185, 1996.
18 T. Kahl. A fibration category of local pospaces. *ENTCS*, 230:129–140, 2009.
19 M. Kelly. *Basic concept of enriched category theory*. CUP, 1982.
20 D. Park. Concurrency and Automata on Infinite Sequences. *TCS*, 104:167–183, 1981.
21 T. Porter. Enriched categories and models for spaces of evolving states. *TCS*, 405:88–100, 2008.

22 T. Porter. Steps towards a 'directed homotopy hypothesis'. $(\infty, 1)$-categories, directed spaces and perhaps rewriting. In *HCR*, 2015. URL: `http://www.lix.polytechnique.fr/Labo/Samuel.Mimram/docs/hcr/porter.pdf`.

23 V. R. Pratt. Modeling Concurrency with Geometry. In *PoPL*, 1991.

24 D. Quillen. *Homotopical Algebra*. Springer, 1965.

25 M. Raussen. Invariants of directed spaces. *ACS*, 15(4):355–386, 2007.

26 A. Strøm. The homotopy category is a homotopy category. *Arc. der Mathematik*, 23, 1972.

27 R. van Glabbeek. Bisimulation for higher-dimensional automata, 1991.

Robust Linear Temporal Logic*

Paulo Tabuada[1] and Daniel Neider[2]

1 **Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA**
tabuada@ucla.edu

2 **Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA**
neider@automata.rwth-aachen.de

Abstract

Although it is widely accepted that every system should be robust, in the sense that "small" violations of environment assumptions should lead to "small" violations of system guarantees, it is less clear how to make this intuitive notion of robustness mathematically precise. In this paper, we address the problem of how to specify robustness in temporal logic. Our solution consists of a robust version of the Linear Temporal Logic (LTL) fragment that only contains the always and eventually temporal operators. We denote this new logic by rLTL($\boxdot, \diamondsuit$). Its formulas are syntactically identical to LTL formulas but are endowed with a many-valued semantics that encodes robustness. In particular, the semantics of the rLTL formula $\varphi \Rightarrow \psi$ is such that a "small" violation of the environment assumption φ is guaranteed to only produce a "small" violation of the system guarantee ψ. In addition, we study the verification and synthesis problems for this logic. Similarly to LTL, we show that: both problems are decidable; the verification problem can be solved in exponential time; the synthesis problem is solvable in doubly exponential time. All the results for rLTL($\boxdot, \diamondsuit$) smoothly extend to full rLTL, the robust version of full LTL. For reasons of space, such extension is not discussed but available in an extended version [21].

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

Keywords and phrases Linear Temporal Logic, Robustness

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.10

1 Introduction

Specifications for open reactive systems are typically written as an implication

$$\varphi \Rightarrow \psi, \tag{1}$$

where φ is an environment assumption and ψ is a system guarantee. In Linear Temporal Logic (LTL), this implication is equivalent to $\neg\varphi \vee \psi$. Hence, whenever the assumption φ is violated the system can behave arbitrarily. This is clearly inadequate since environment assumptions will *inevitably* be violated: the real environment where the system will be deployed is not completely known at design time and thus cannot be accurately described by the formula φ.

We argue that a robust design satisfies the implication in (1) in a robust manner (i.e., a "small" violation of φ results, at most, in a "small" violation of ψ). To make this intuitive

* This research was partially supported by the NSF project ExCAPE: Expeditions in Computer Augmented Program Engineering (CCF-1138996).

© Paulo Tabuada and Daniel Neider;
licensed under Creative Commons License CC-BY

notion of *robustness* mathematically precise, we introduce a fragment of a new logic termed *robust Linear Temporal Logic* and simply denoted by rLTL. For reasons of space, we only discuss the fragment of rLTL that contains the always and eventually temporal operators, denoted by rLTL($\boxdot, \diamondsuit$). However, all the decidability and complexity results extend to full rLTL as discussed in [21]. In developing rLTL, we were guided by two objectives: 1) the syntax of rLTL should be similar to the syntax of LTL in order to make the transition from LTL to rLTL as transparent as possible; 2) robustness should be intrinsic to the logic rather than extrinsic (i.e., robustness should not rely on the ability of the designer to provide quantitative information such as ranks, costs, or quantitative interpretations of atomic propositions). This guarantees that verification and synthesis techniques for rLTL are widely applicable as they *only* require an LTL specification.

The main conceptual question to be addressed when developing the semantics of rLTL is how to give mathematical meaning to "small" violations of a formula φ. The approach advocated in this paper can be intuitively explained by regarding LTL formulas of the form $\Box p$, $\diamondsuit\Box p$, $\Box\diamondsuit p$, and $\diamondsuit p$, for an atomic proposition p, as requirements on the number of times that p is satisfied over time. Under this interpretation, and for the formula $\varphi = \Box p$, there is a clear ordering among the possible temporal evolutions of p: p being satisfied at every time instant is preferred to p being violated at finitely many time instants which, in turn, is preferred to p being satisfied and violated at infinitely many time instants. The latter case is preferred to p only being satisfied at finitely many time instants and this case is preferred over p being satisfied at no time instant. A semantics that would distinguish between these different five cases would then enable us to state that violating $\Box p$ while satisfying $\diamondsuit\Box p$ consists of a smaller violation of the formula $\varphi = \Box p$ than violating $\Box p$ while satisfying $\Box\diamondsuit p$. Making these ideas mathematically rigorous requires a 5-valued semantics that we develop in this paper. Interestingly, our specific interpretation of the five different truth values leads to an intuitionistic semantics where negation is dualized and to a corresponding algebraic structure, *da Costa algebras*, which were only very recently investigated [18].

Contributions

The first contribution of this paper (Section 3) is the fragment rLTL($\boxdot, \diamondsuit$) of our new logic rLTL, which enables reasoning about robustness of LTL specifications. The syntax of rLTL($\boxdot, \diamondsuit$) is almost identical to the syntax of LTL with the only notable difference being dotted temporal operators. The 5-valued semantics of rLTL($\boxdot, \diamondsuit$) is, however, different in many regards (e.g., it precludes several LTL identities to hold on rLTL). For that reason, we carefully motivate the need for a many-valued semantics and provide several examples illustrating how rLTL can be used to reason about robustness.

The second contribution (Section 4) is the study of several computational questions related to rLTL($\boxdot, \diamondsuit$). We first show that rLTL($\boxdot, \diamondsuit$) is as expressive as the LTL($\Box, \diamondsuit$) fragment by providing effective translations between both logics. However, the translation from rLTL($\boxdot, \diamondsuit$) to LTL($\Box, \diamondsuit$) involves an exponential blow-up, leaving open the possibility of improved complexity bounds for the rLTL($\boxdot, \diamondsuit$) verification and synthesis problems. Indeed, the exponential blow-up can be avoided by a carefully generalization of the construction that associates with each LTL formula φ a Büchi automaton $\mathcal{A}_\varphi$ recognizing all infinite words satisfying φ. Critical to this new construction are the properties of the da Costa algebra, used to define the rLTL($\boxdot, \diamondsuit$) semantics, which can be leveraged to keep the size of $\mathcal{A}_\varphi$ in $\mathcal{O}(|\mathrm{cl}(\varphi)| \cdot 5^{|\mathrm{cl}(\varphi)|})$ where $\mathrm{cl}(\varphi)$ denotes the set of subformulas of φ. Note that this is the same complexity bound as for LTL where we replace 2 (LTL has a 2-valued semantics) with

5 (rLTL($\boxdot, \diamondsuit$) has a 5-valued semantics). Additional consequences of the construction of $\mathcal{A}_\varphi$ include: 1) rLTL($\boxdot, \diamondsuit$) specifications can be verified in time exponential in the size of specification and polynomial in the size of the system being verified; 2) the time complexity of synthesizing reactive controllers for rLTL($\boxdot, \diamondsuit$) specifications is doubly exponential in the size of the specification and polynomial in the size of the underlying game graph.

Finally, we show in Section 5 that by dualizing the semantics of rLTL in a specific sense one obtains a logic that is adequate to reason about quality.

Related efforts

Several efforts to robustify the implication in (1) have been reported in the literature. Bloem et al. [5] formalized robustness by comparing how often the system violates its assumptions with how often the environment violates its assumptions. Counting the number of violations requires the designer to provide quantitative information in the form of error functions. In contrast, when working with rLTL, the designer only needs to provide an LTL specification. A different notion of robustness appeared in [8] and requires the effect of a sporadic disturbance to disappear in finite time. The semantics of rLTL was built so as to naturally encode this as well as other requirements expressing how a weakening of the system assumptions should lead to a weakening of the system guarantees. Previous work by one of the authors, reported in [22], provided a single notion of robustness encompassing the notion in [8] but requiring the designer to provide quantitative information in the form of a cost. Such cost implicitly specifies how guarantees and assumptions are to be weakened in a robust design and was inspired by the work of Alur et al. [2]. A different formalization of robustness appeared in [9], which considered a specific class of violations of safety assumptions defined by the frequency of violations. In contrast to all the previously described approaches, the results in this paper do not require any additional assumptions or input from a designer beyond an LTL formula.

Robustness for liveness specifications was discussed by Bloem et al. [4], who considered specifications of the form $\wedge_{i\in I}\varphi_i \Rightarrow \wedge_{j\in J}\psi_j$, where φ_i and ψ_j are formulas of the form $\diamondsuit\square p$ for some atomic proposition p (depending on i and j). Robustness is then measured by comparing the number of violated environment assumptions φ_i with the number of violated guarantees ψ_j. This approach is incomparable with ours since the rLTL($\boxdot, \diamondsuit$) semantics does not distinguish between the violation of one or multiple assumptions. It does, however, distinguish between the different ways in which φ_i and ψ_j can be violated. Although robustness is formalized differently, rLTL($\boxdot, \diamondsuit$) can be used to reason about the robustness of both safety and liveness specifications. Also incomparable with rLTL($\boxdot, \diamondsuit$) is the work by Chaudhuri et al. [6] and by Majumdar et al. [14], which considers continuity properties of software expressed by the requirement that a deviation in a program's input causes a proportional deviation in its output. Although natural, these notions of robustness only apply to the Turing model of computation and not to the reactive model employed in this paper. Robustness was also investigated in the context of systems biology (e.g., Rizk et al. [19] and Češka et al. [23]) although the considered models are quite different as they require continuity and stochasticity.

There exists a large body of work on many-valued logics that we will not attempt to review here since it does not directly address questions of robustness. We do, however, allow for one exception: the work by Fainekos and Pappas [10] on robustness of temporal logic over continuous signals and its extensions (e.g., by Donzé and Maler [7]). These results, however, require continuous-valued signals whereas rLTL is to be used in the classical setting of discrete-time and finite valued signals (e.g., as in transition systems).

The last body of work related to the contents of this paper is the work on lattice automata and lattice LTL [13, 1]. The syntax of lattice LTL is similar to the syntax of LTL except that

atomic propositions assume values on a finite lattice. Lattice LTL derives its many-valued character from the atomic propositions, whereas atomic propositions in rLTL($\boxdot, \diamondsuit$) are interpreted classically (i.e., they only assume two truth values). Therefore, the many-valued character of rLTL($\boxdot, \diamondsuit$) arises from the temporal evolution of the atomic propositions and not from the nature of the atomic propositions or their interpretation. In fact, if we only allow two truth values for the atomic propositions in lattice LTL, this logic specializes to LTL. Hence, these two logics capture orthogonal considerations, and results on lattice LTL and lattice automata do not shed light on how to address similar problems for rLTL($\boxdot, \diamondsuit$).

2 Notations and Review of LTL

Let $\mathbb{N} = \{0, 1, \ldots\}$ be the set of natural numbers and $\mathbb{B} = \{0, 1\}$ be the set of Boolean values (0 interpreted as *false* and 1 as *true*). For a set S, let 2^S be the *powerset* of S and S^ω be the set of all *infinite sequences* of elements of S. An *alphabet*, usually denoted by Σ, is a finite, nonempty set whose elements are called *symbols*. An infinite sequence $\sigma = a_0 a_1 \ldots$ of symbols with $a_i \in \Sigma$, $i \in \mathbb{N}$, is called an *infinite word*. For an infinite word $\sigma = a_0 a_1 \ldots \in \Sigma^\omega$ and $i \in \mathbb{N}$, let $\sigma(i) = a_i$ denote the i-th symbol of σ and $\sigma_{i..}$ the (infinite) suffix of σ starting at position i (i.e., $\sigma_{i..} = \sigma_i \sigma_{i+1} \ldots \in \Sigma^\omega$). In particular, we have the equality $\sigma_{0..} = \sigma$.

Next, we recapitulate the syntax and semantics of a fragment of *Linear Temporal Logic (LTL)*, abbreviated as LTL($\Box, \diamondsuit$), that is restricted to the always and eventually modalities.

▶ **Definition 1** (LTL($\Box, \diamondsuit$) syntax). Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. *LTL*($\Box, \diamondsuit$) *formulas* are inductively defined as follows: (a) each $p \in \mathcal{P}$ is an LTL formula and (b) if φ and ψ are LTL formulas, so are $\neg\varphi$, $\varphi \vee \psi$, $\Box\varphi$, and $\diamondsuit\varphi$.

We also allow the formulas *true*, *false*, $\varphi \wedge \psi$, and $\varphi \Rightarrow \psi$ with their usual meaning.

In the following, we define the semantics of LTL($\Box, \diamondsuit$) by a mapping W that maps an infinite word $\sigma \in \Sigma^\omega$, $\Sigma = 2^{\mathcal{P}}$, and an LTL($\Box, \diamondsuit$) formula φ to the element $W(\sigma, \varphi) \in \mathbb{B}$. Although the semantics of LTL is usually defined by a satisfaction relation, we here define it in terms of a function so as to be consistent with our definition of the rLTL($\boxdot, \diamondsuit$) semantics.

▶ **Definition 2** (LTL($\Box, \diamondsuit$) semantics). The *LTL*($\Box, \diamondsuit$) *semantics* is a mapping W, called *valuation*, defined as follows:

$$W(\sigma, p) = \begin{cases} 0 & p \notin \sigma(0); \text{ and} \\ 1 & p \in \sigma(0) \end{cases} \quad (2)$$

$$W(\sigma, \neg\varphi) = 1 - W(\sigma, \varphi) \quad (4)$$

$$W(\sigma, \Box\varphi) = \inf_{i \geq 0} W(\sigma_{i..}, \varphi) \quad (5)$$

$$W(\sigma, \varphi \vee \psi) = \max\{W(\sigma, \varphi), W(\sigma, \psi)\} \quad (3)$$

$$W(\sigma, \diamondsuit\varphi) = \sup_{i \geq 0} W(\sigma_{i..}, \varphi) \quad (6)$$

We often use a compact notation when referring to infinite words over sets of atomic propositions: instead of writing the set of atomic propositions corresponding to a symbol, we use simple propositional formulas, such as p, $\neg p$, and $p \wedge q$, to denote all the sets of atomic propositions where these formulas hold true according to the LTL semantics. For instance, given $\mathcal{P} = \{p, q, r\}$, we write $\neg p$ to denote the sets $\emptyset, \{q\}, \{r\}, \{q, r\} \in \Sigma$, and we write $p \wedge q$ to denote the sets $\{p, q\}, \{p, q, r\} \in \Sigma$.

3 The Syntax and Semantics of rLTL($\boxdot, \diamondsuit$)

In this section we introduce the fragment rLTL($\boxdot, \diamondsuit$) that intrinsically provides a robust interpretation of LTL($\Box, \diamondsuit$) formulas. On the one hand, rLTL($\boxdot, \diamondsuit$) is simple enough that we can provide a lucid intuitive explanation for its semantics. On the other hand,

rLTL($\boxdot, \diamondsuit$) already illustrates most of the technical difficulties brought by robustness. Full rLTL, including all technical results, is presented in [21].

3.1 The Syntax of rLTL($\boxdot, \diamondsuit$)

The syntax of rLTL($\boxdot, \diamondsuit$) closely mirrors the syntax of LTL($\Box, \diamondsuit$) with the only noticeable difference being the use of dotted temporal operators.

▶ **Definition 3** (rLTL($\boxdot, \diamondsuit$) syntax)**.** Let $\mathcal{P}$ be a nonempty, finite set of atomic propositions. *rLTL*($\boxdot, \diamondsuit$) *formulas* are inductively defined as follows: (a) each $p \in \mathcal{P}$ is an rLTL formula and (b) if φ and ψ are rLTL formulas, so are $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$, $\boxdot\varphi$, and $\diamondsuit\varphi$.

In LTL, we can derive the conjunction and implication operators from negation and disjunction. This is no longer the case in rLTL($\boxdot, \diamondsuit$) since it has a many-valued semantics. For this reason, conjunction and implication occur explicitly in Definition 3.

3.2 Robustness and Counting

Consider the LTL formula $\Box p$ where p is an atomic proposition. There is only one way in which this formula can be satisfied: p holds at every time step. In contrast, there are several ways in which this formula can be violated, and we seek a semantics that distinguishes them.

It seems intuitively clear to the authors that the worst manner in which $\Box p$ fails to be satisfied occurs when p fails to hold at every time step. Although still violating $\Box p$, we would prefer a situation where p holds for at most finitely many time instants. Better yet would be that p holds at infinitely many instants while it fails to hold also at infinitely many instants. Finally, among all the possible ways in which $\Box p$ can be violated, we would prefer the case where p fails to hold for at most finitely many time instants. Consequently, our robust semantics is designed to distinguish between satisfaction and these four possible different ways to violate $\Box p$. However, as convincing as this argument might be, a question persists: in which sense can we regard these five alternatives as canonical?

We answer this question by interpreting satisfaction of $\Box p$ as a counting problem. The previously discussed five different cases, satisfaction and four different types of violation, can be seen as the result of counting the number of 0s and 1s in the word $\alpha = W(\sigma_{0..}, p)W(\sigma_{1..}, p) \ldots \in \mathbb{B}^\omega$ rather than using the inf-operator in (5). From this perspective, satisfaction corresponds to the number of occurrences of 0 being zero. Among all the possible ways in which $\Box p$ can be violated, the most preferred occurs when p only fails to hold at finitely many time instants. This corresponds to a finite number of 0s in α. All the other ways in which $\Box p$ can be violated are similarly identified by the number of 0s and 1s in α.

We say that an LTL($\Box, \diamondsuit$) formula φ is a counting formula if its valuation $W(\sigma, \varphi)$ only depends on the number of occurrences of each atomic proposition but not on its order. Such formula φ is essentially counting how many times each atomic proposition appears along the word σ. Formally, we say that φ is a counting formula if for every infinite word $\sigma \in \Sigma^\omega$, seen as a map $\sigma\colon \mathbb{N} \to \Sigma$, and for every bijection $f\colon \mathbb{N} \to \mathbb{N}$ we have $W(\sigma, \varphi) = W(\sigma \circ f, \varphi)$. Recall that by composing a sequence of permutations (bijections) one again obtains a bijection. Hence, by permuting the elements of σ, we obtain the word $\sigma \circ f$ where f is the composition of the employed permutations. If we now assume $\mathcal{P} = \{p\}$, then we can always permute the elements of σ so that the permuted word $\sigma \circ f$ is of the form $(\neg p\, p)^k p^\omega$, $(\neg p\, p)^\omega$, or $(\neg p\, p)^k (\neg p)^\omega$, where $k \in \mathbb{N}$. We further recall that formulas in LTL($\Box, \diamondsuit$) can only define

stutter-invariant properties [15]. Therefore, the semantics of LTL($\Box, \Diamond$) cannot distinguish[1] between the words $(\neg p\, p)^{k_1}\, p^\omega$ and $(\neg p\, p)^{k_2}\, p^\omega$ for $k_1 \neq k_2$, and $k_1, k_2 > 0$, although it can distinguish between the case $k_1 = 0$ and $k_2 > 0$. The same argument applies to the words $(\neg p\, p)^{k_1}\, (\neg p)^\omega$ and $(\neg p\, p)^{k_2}\, (\neg p)^\omega$ and shows that there are only five canonical forms that can be distinguished by LTL($\Box, \Diamond$):

$$p^\omega, (\neg p\, p)^+\, p^\omega, (\neg p\, p)^\omega, (\neg p\, p)^+\, (\neg p)^\omega, \text{ and } (\neg p)^\omega. \tag{7}$$

It should be no surprise that these are exactly the five cases we previously discussed.

The considerations in this section suggest the need for a semantics that is 5-valued rather than 2-valued so that we can distinguish between the aforementioned five cases. Therefore, we need to replace Boolean algebras by a different type of algebraic structure that can accommodate a 5-valued semantics for each rLTL($\boxdot, \diamondsuit$) formula.

3.3 da Costa Algebras

According to our motivating example $\Box p$, the desired semantics should have one truth value corresponding to *true* and four truth values corresponding to different shades of *false*. It is instructive to think of truth values as the elements of $\mathbb{B}^4$ (i.e., the four-fold Cartesian product of $\mathbb{B}$) that arise as the possible values of the 4-tuple of LTL formulas:

$$(\Box p, \Diamond\Box p, \Box\Diamond p, \Diamond p). \tag{8}$$

To ease notation, we denote such values interchangeably by $b = b_1b_2b_3b_4$ and $b = (b_1, b_2, b_3, b_4)$ with $b_i \in \mathbb{B}$ for $i \in \{1, 2, 3, 4\}$. The value 1111 then corresponds to *true* since $\Box p$ is satisfied. The most preferred violation of $\Box p$ (p fails to hold at only finitely many time instants) corresponds to 0111, followed by 0011 (p holds at infinitely many instants and also fails to hold at infinitely many instants), 0001 (p holds at most at finitely many instants), and 0000 (p fails to hold at every time instant). Such preferences can be encoded in the linear order

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111 \tag{9}$$

that renders the set $\mathbb{B}_4 = \{0000, 0001, 0011, 0111, 1111\}$ a (bounded) distributive lattice with top element $\top = 1111$ and bottom element $\bot = 0000$. In $\mathbb{B}_4$, the meet $\sqcap$ can be interpreted as minimum and the join $\sqcup$ as maximum with respect to the order in (9). We use $\sqcap$ and $\sqcup$ when discussing lattices in general and use min and max for the specific lattice $\mathbb{B}_4$ or the Boolean algebra $\mathbb{B}$.

The first choice to be made in using the lattice $\mathbb{B}_4$ to define the semantics of rLTL($\boxdot, \diamondsuit$) is the choice of an operation on $\mathbb{B}_4$ modeling conjunction. Consider the formula $\Box p \wedge \Box q$ and the word $\sigma = \neg(p \wedge q)(p \wedge q)^\omega$. As introduced above, the value of $\Box p$ on σ corresponds to 0111 and the value of $\Box q$ on σ corresponds to 0111 since on both cases we have the most preferred violation of the formulas. Therefore, the value of $\Box p \wedge \Box q$ on σ should also be 0111 since the formula $\Box p \wedge \Box q$ is only violated a finite number of times. It thus seems natural[2]

---

[1] To see why this is the case, note that any word $(\neg p\, p)^k\, p^\omega$ with $k \in \mathbb{N}$ can be permuted to the form $(\neg p)^k\, p^k p^\omega$ and by stutter invariance can be reduced to $\neg p\, p\, p^\omega$.

[2] Note that there are situations where it is convenient to model conjunction differently. In the related work, we referenced the work of Bloem et al. [4], where the specific way in which robustness is modeled requires distinguishing between the number of conjuncts that are satisfied in the assumption $\wedge_{i \in I} \varphi_i$. This cannot be accomplished if conjunction is modeled by min and a different triangular-norm would have to be used for this purpose. Note that both Łukasiewicz's conjunction as well as Goguen's conjunction have the property that their value decreases as the number of conjuncts that are true decreases.

to model conjunction in $\mathbb{B}_4$ by min and, for similar reasons, to model disjunction in $\mathbb{B}_4$ by max.

As in intuitionistic logic[3], our implication is defined as the residue of $\sqcap$. In other words, we define the implication $a \to b$ by requiring that $c \preceq a \to b$ if and only if $c \sqcap a \preceq b$ for every $c \in \mathbb{B}_4$. This leads to

$$a \to b = \begin{cases} 1111 & \text{if } a \preceq b; \text{ and} \\ b & \text{otherwise.} \end{cases}$$

However, we now *diverge* from intuitionistic logic (and most many-valued logics) where negation of a is defined by $a \to 0000$. Such negation is not compatible with the interpretation that all the elements of $\mathbb{B}_4$, except for 1111, represent (different shades of) *false* and thus their negation should have the truth value 1111. To make this point clear, the table below presents the intuitionistic negation in $\mathbb{B}_4$ and the desired negation compatible with our interpretation of the truth values in $\mathbb{B}_4$.

Value	Desired negation	Intuitionistic negation
1111	0000	0000
0111	1111	0000
0011	1111	0000
0001	1111	0000
0000	1111	1111

What is then the algebraic structure on $\mathbb{B}_4$ that supports the desired negation? This very same problem was recently investigated by Priest [18], and the answer is *da Costa* algebras.

▶ **Definition 4** (da Costa algebra). A *da Costa* algebra is a 6-tuple $(A, \sqcap, \sqcup, \preceq, \to, \overline{\cdot})$ where

1. $(A, \sqcap, \sqcup, \preceq)$ is a distributive lattice where $\preceq$ is the ordering derived from $\sqcap$ and $\sqcup$;
2. $\to$ is the residual of $\sqcap$ (i.e., $a \preceq b \to c$ if and only if $a \sqcap b \preceq c$ for every $a, b, c \in A$);
3. $a \preceq b \sqcup \overline{b}$ for every $a, b \in A$; and
4. $\overline{a} \preceq b$ whenever $c \sqcup \overline{c} \preceq a \sqcup b$ for every $a, b, c \in A$.

Indeed, $\mathbb{B}_4$ is a da Costa algebra if we use the desired negation defined in the table above.

It should be mentioned that working with a 5-valued semantics has its price. The law of non-contradiction fails in $\mathbb{B}_4$ (i.e., $a \sqcap \overline{a}$ may not equal $\bot = 0000$ as evidenced by taking $a = 0111$). However, since $a \sqcap \overline{a} \prec 1111$, a weak form of non-contradiction still holds as $a \sqcap \overline{a}$ is to be interpreted as a shade of *false* but not necessarily as the least preferred way of violating $a \sqcap \overline{a}$, which corresponds to $\bot$. Contrary to intuitionistic logic, the law of excluded middle is valid (i.e., $a \sqcup \overline{a} = \top = 1111$). Finally, $a = 0111$ shows that $\overline{\overline{a}} \neq a$ although it is still true that $\overline{\overline{a}} \to a$.

[3] This is also done in context of residuated lattices that is more general than the Heyting algebras used in intuitionistic logic. Recall that a residuated lattice is a lattice $(A, \sqcap, \sqcup)$, satisfying same additional conditions, and equipped with a commutative monoid $(A, \otimes, \mathbf{1})$ satisfying some additional compatibility conditions. Since we chose the lattice meet $\sqcap$ to represent conjunction, we have a residuated lattice where $\otimes = \sqcap$ and $\mathbf{1} = \top$.

3.4 Semantics of rLTL($\boxdot, \diamondsuit$) on da Costa Algebras

The semantics of rLTL($\boxdot, \diamondsuit$) is given by a mapping V, called *valuation* as in the case of LTL($\Box, \diamondsuit$), that maps an infinite word $\sigma \in \Sigma^\omega$ and an rLTL($\boxdot, \diamondsuit$) formula φ to an element of $\mathbb{B}_4$. In defining V, we judiciously use the algebraic operations of the da Costa algebra $\mathbb{B}_4$ to give meaning to the logical connectives in the syntax of rLTL($\boxdot, \diamondsuit$). In the following, let $\Sigma = 2^{\mathcal{P}}$ for a finite set of atomic propositions $\mathcal{P}$.

On atomic propositions $p \in \mathcal{P}$, V is defined "classically" by

$$V(\sigma, p) = \begin{cases} 0000 & \text{if } p \notin \sigma(0); \text{ and} \\ 1111 & \text{if } p \in \sigma(0). \end{cases} \tag{10}$$

Since we are using a 5-valued semantics, we provide a separate definition for all the four logical connectives:

$$V(\sigma, \varphi \wedge \psi) = V(\sigma, \varphi) \sqcap V(\sigma, \psi) \tag{11}$$

$$V(\sigma, \neg\varphi) = \overline{V(\sigma, \varphi)} \tag{12}$$

$$V(\sigma, \varphi \vee \psi) = V(\sigma, \varphi) \sqcup V(\sigma, \psi) \tag{13}$$

$$V(\sigma, \varphi \Rightarrow \psi) = V(\sigma, \varphi) \rightarrow V(\sigma, \psi) \tag{14}$$

Note how the semantics mirrors the algebraic structure of da Costa algebras. This is no accident since valuations are typically algebra homomorphisms. Unfortunately, da Costa algebras are not equipped[4] with operations corresponding to $\boxdot$ and $\diamondsuit$, the robust versions of $\Box$ and $\diamondsuit$, respectively. Therefore, we resort to the counting interpretation in Section 3.2 to motivate the semantics of $\boxdot$. Formally, we have

$$V(\sigma, \boxdot\varphi) = \big(\inf_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{j\geq 0}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi), \inf_{j\geq 0}\sup_{i\geq j} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\big) \tag{15}$$

where $V_k(\sigma, \varphi) = \pi_k \circ V(\sigma, \varphi)$ for $k \in \{1, 2, 3, 4\}$ and $\pi_k : \mathbb{B}_4 \to \mathbb{B}$ are mappings defined by

$$\pi_k(a_1, a_2, a_3, a_4) = a_k. \tag{16}$$

To illustrate the semantics of $\boxdot$, consider the simple case where φ is an atomic proposition p. This means that one can express $V(\sigma, \boxdot p)$ in terms of the LTL valuation W by

$$V(\sigma, \boxdot p) = \big(W(\sigma, \Box p), W(\sigma, \diamondsuit\Box p), W(\sigma, \Box\diamondsuit p), W(\sigma, \diamondsuit p)\big) \tag{17}$$

thus connecting the semantics of $\boxdot$ to the counting problems described in Section 3.2 and to the 4-tuple of LTL formulas in (8).

The last operator is $\diamondsuit$ and its semantics is given by

$$V(\sigma, \diamondsuit\varphi) = \big(\sup_{i\geq 0} V_1(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_2(\sigma_{i..}, \varphi), \sup_{i\geq 0} V_3(\sigma_{i..}, \varphi),\ \sup_{i\geq 0} V_4(\sigma_{i..}, \varphi)\big). \tag{18}$$

According to the counting problems of Section 3.2, there is only one way in which the LTL formula $\diamondsuit p$, for an atomic proposition p, can be violated, namely p never holds. Hence, $V(\sigma, \diamondsuit\varphi)$ is one of only two possible truth values: 1111 or 0000. We further note that $\diamondsuit$ is not dual to $\boxdot$, as expected in a many-valued logic.

Having defined the semantics of rLTL($\boxdot, \diamondsuit$), let us now see if the formula $\boxdot p \Rightarrow \boxdot q$, where $\boxdot p$ is an environment assumption and $\boxdot q$ is a system guarantee with $p, q \in \mathcal{P}$, lives to the expectations set in the introduction and to the intuition provided in Section 3.2.

[4] One could consider developing a notion of da Costa algebras with operators in the spirit of Boolean algebras with operators [12]. We leave such investigation for future work.

1. According to (17), if $\Box p$ holds, then $\boxdot p$ evaluates to 1111 and the implication $\boxdot p \Rightarrow \boxdot q$ is *true* (i.e., the value of $\boxdot p \Rightarrow \boxdot q$ is 1111) if $\boxdot q$ evaluates to 1111 (i.e., if $\Box q$ holds). Hence, the desired behavior of $\Box p \Rightarrow \Box q$, when the environment assumptions hold, is retained.
2. Consider now the case where $\Box p$ fails but the weaker assumption $\Diamond\Box p$ holds. In this case $\boxdot p$ evaluates to 0111 and the implication $\boxdot p \Rightarrow \boxdot q$ is *true* if $\boxdot q$ evaluates to 0111 or higher. This means that $\Diamond\Box q$ needs to hold.
3. A similar argument shows that we can also conclude the following consequences whenever $\boxdot p \Rightarrow \boxdot q$ evaluates to 1111: $\Box\Diamond q$ follows whenever the environment satisfies $\Box\Diamond p$ and $\Diamond q$ follows whenever the environment satisfies $\Diamond p$.

We thus conclude that the semantics of $\boxdot p \Rightarrow \boxdot q$ captures the desired robustness property by which a weakening of the assumption $\boxdot p$ leads to a weakening of the guarantee $\boxdot q$. The following example further motivates the usefulness of the proposed semantics.

3.5 Examples

The usefulness of implications that are not true: We argued in the previous section that rLTL($\boxdot, \diamondsuit$) captures the intended robustness properties for the specification $\boxdot p \Rightarrow \boxdot q$ whenever this formula is *true* (i.e., evaluates to 1111). But does the formula $\boxdot p \Rightarrow \boxdot q$ still provide useful information when its valuation is lower than 1111? It follows from the semantics of implication that $V(\sigma, \boxdot p \Rightarrow \boxdot q) = b$, for $b \prec 1111$, occurs when $V(\sigma, \boxdot q) = b$ (i.e., whenever a value of b can be guaranteed despite b being smaller than $V(\sigma, \boxdot p)$). The value $V(\sigma, \boxdot p \Rightarrow \boxdot q)$ thus describes which weakened guarantee follows from the environment assumption whenever the intended system guarantee does not. This can be seen as another measure of robustness: despite $\boxdot q$ not following from $\boxdot p$, the behavior of the system is not arbitrary, a value of b is still guaranteed.

Non-counting formulas: Consider the non-counting formula $\Box(p \Rightarrow \Diamond q)$, which requires each occurrence of p to be followed by an occurrence of q. This formula is routinely used in the literature to illustrate LTL and, hence, constitutes a litmus test to rLTL($\boxdot, \diamondsuit$). The semantics of the dotted version of $\Box(p \Rightarrow \Diamond q)$ can be expressed using the LTL valuation as

$$V(\sigma, \boxdot(p \Rightarrow \diamondsuit q)) = \\ \big(W(\sigma, \Box(p \Rightarrow \Diamond q)), W(\sigma, \Box\Diamond p \Rightarrow \Box\Diamond q), W(\sigma, \Diamond\Box p \Rightarrow \Box\Diamond q), W(\sigma, \Box p \Rightarrow \Diamond q)\big).$$

It is interesting to observe how the semantics of $\varphi = \boxdot(p \Rightarrow \diamondsuit q)$ recovers: 1) strong fairness, also known as compassion, when the value of φ is 0111; 2) weak fairness, also known as justice, when the value of φ is 0011; and 3) the even weaker notion of fairness described by the formula $\Box p \Rightarrow \Diamond q$, when the value of φ is 0001. The fact that all these different and well known notions of fairness naturally appear in the proposed semantics is another strong indication of rLTL's naturalness and usefulness.

3.6 Relating LTL($\Box, \Diamond$) and rLTL($\boxdot, \diamondsuit$)

In this section we discuss, at the technical level, the relationships between rLTL($\boxdot, \diamondsuit$) and LTL($\Box, \Diamond$).

First, we argue that the semantics of LTL($\Box, \Diamond$) can always be recovered from the first component of the semantics of rLTL($\boxdot, \diamondsuit$). To this end, recall the mapping $\pi_1 \colon \mathbb{B}_4 \to \mathbb{B}$ introduced in (16), defined by $\pi_1(a_1, a_2, a_3, a_4) = a_1$. Composing π_1 with a valuation V

of rLTL($\boxdot, \diamondsuit$), we obtain the function $V_1 = \pi_1 \circ V$ mapping an infinite word $\sigma \in \Sigma^\omega$ and a rLTL($\boxdot, \diamondsuit$) formula φ to a Boolean value. In fact, V_1 corresponds to the LTL($\square, \diamondsuit$) semantics, which implies that rLTL($\boxdot, \diamondsuit$) is as expressive as LTL($\square, \diamondsuit$). The proof of this claim is a straightforward case distinction. On atomic propositions $p \in \mathcal{P}$ we have

$$V_1(\sigma, p) = \begin{cases} \pi_1(0000) = 0 & \text{if } p \notin \sigma(0); \text{ and} \\ \pi_1(1111) = 1 & \text{if } p \in \sigma(0). \end{cases}$$

Moreover, the following holds for "Boolean" connectives:

$$\begin{aligned} V_1(\sigma, \varphi \wedge \psi) &= \pi_1\left(V(\sigma, \varphi) \sqcap V(\sigma, \psi)\right) = \min\{V_1(\sigma, \varphi), V_1(\sigma, \psi)\}, \\ V_1(\sigma, \varphi \vee \psi) &= \pi_1\left(V(\sigma, \varphi) \sqcup V(\sigma, \psi)\right) = \max\{V_1(\sigma, \varphi), V_1(\sigma, \psi)\}, \\ V_1(\sigma, \neg\varphi) &= \pi_1(\overline{V(\sigma, \varphi)}) = 1 - \pi_1\left(V(\sigma, \varphi)\right) = 1 - V_1(\sigma, \varphi), \\ V_1(\sigma, \varphi \Rightarrow \psi) &= \pi_1\left(V(\sigma, \varphi) \rightarrow V(\sigma, \psi)\right) = \max\left\{1 - V_1(\sigma, \varphi), V_1(\sigma, \psi)\right\}. \end{aligned}$$

Finally, the following follows directly from the semantics of $\boxdot$ and $\diamondsuit$:

$$\begin{aligned} V_1(\sigma, \boxdot\varphi) &= \pi_1\left(V(\sigma, \boxdot\varphi)\right) = \inf_{i \geq 0} V_1(\sigma_{i..}, \varphi), \\ V_1(\sigma, \diamondsuit\varphi) &= \pi_1\left(V(\sigma, \diamondsuit\varphi)\right) = \sup_{i \geq 0} V_1(\sigma_{i..}, \varphi). \end{aligned}$$

It is not hard to verify that V_1 in all cases corresponds to the LTL($\square, \diamondsuit$) valuation.

Conversely, one can translate an rLTL($\boxdot, \diamondsuit$) formula φ into four LTL($\square, \diamondsuit$) formulas $\psi_\varphi^1, \ldots, \psi_\varphi^4$ such that

$$\pi_j(V(\sigma, \varphi)) = V_j(\sigma, \varphi) = W(\sigma, \psi_\varphi^j)$$

for all $\sigma \in \Sigma^\omega$ and $j \in \{1, \ldots, 4\}$. The key idea is to emulate the semantics of each operator occurring in φ component-wise by means of dedicated LTL formulas.

The construction of ψ_φ^j proceeds by induction over the subformulas of φ:

- If $\varphi = p$ for an atomic proposition $p \in \mathcal{P}$, then $\psi_\varphi^j \coloneqq p$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \vee \varphi_2$, then $\psi_\varphi^j \coloneqq \psi_{\varphi_1}^j \vee \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \wedge \varphi_2$, then $\psi_\varphi^j \coloneqq \psi_{\varphi_1}^j \wedge \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \neg\varphi_1$, then $\psi_\varphi^j \coloneqq \neg(\psi_{\varphi_1}^1 \wedge \psi_{\varphi_1}^2 \wedge \psi_{\varphi_1}^3 \wedge \psi_{\varphi_1}^4)$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \varphi_1 \Rightarrow \varphi_2$, then $\psi_\varphi^j \coloneqq \left(\bigvee_{k \in \{1,\ldots,4\}} (\psi_{\varphi_1}^k \wedge \neg\psi_{\varphi_2}^k)\right) \Rightarrow \psi_{\varphi_2}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \diamondsuit\varphi_1$, then $\psi_\varphi^j \coloneqq \diamondsuit\psi_{\varphi_1}^j$ for all $j \in \{1, \ldots, 4\}$.
- If $\varphi = \boxdot\varphi_1$, then $\psi_\varphi^1 \coloneqq \square\psi_{\varphi_1}^1$, $\psi_\varphi^2 \coloneqq \diamondsuit\square\psi_{\varphi_1}^2$, $\psi_\varphi^3 \coloneqq \square\diamondsuit\psi_{\varphi_1}^3$, and $\psi_\varphi^4 \coloneqq \diamondsuit\psi_{\varphi_1}^4$.

It is not hard to verify that the formulas ψ_φ^j have indeed the desired meaning. However, note that the size of ψ_φ^j, measured in the number of subformulas, is exponential in the size of φ due to the recursive substitution of the sub-formulas.

The preceding discussion can be summarized by the following result.

▶ **Proposition 5.** *LTL($\square, \diamondsuit$) and rLTL($\boxdot, \diamondsuit$) are equally expressive.*

Since the translations from LTL($\square, \diamondsuit$) to rLTL($\boxdot, \diamondsuit$) and vice versa are effective, we conclude that any problem for rLTL($\boxdot, \diamondsuit$), whose corresponding problem for LTL($\square, \diamondsuit$) is decidable, is also decidable. In practice, however, the translation from rLTL($\boxdot, \diamondsuit$) to LTL($\square, \diamondsuit$) involves an exponential blow-up. Hence, we now investigate the complexity of several verification and synthesis problems by developing specialized algorithms.

4 Model Checking and Synthesis

Similarly to LTL, rLTL($\boxdot, \diamondsuit$) gives rise to various (decision) problems, some of which we investigate in this section. We are particularly interested in model checking and in reactive synthesis. These two problems are clearly amongst the most important in the context of LTL and, hence, must be investigated for rLTL($\boxdot, \diamondsuit$).

As the translation from rLTL($\boxdot, \diamondsuit$) into LTL($\Box, \Diamond$) potentially results in an exponentially large formula, we now develop a computationally more efficient approach to the model checking and reactive synthesis problems via a translation into (generalized) Büchi automata. Our construction follows the well known translation of LTL into Büchi automata (see, e.g., Baier and Katoen [3]) and results in a generalized Büchi automaton with $\mathcal{O}(k \cdot 5^k)$ states where k counts the subformulas of the given rLTL($\boxdot, \diamondsuit$) formula. This is the same complexity as for the LTL translation – which results in an automation with size in $\mathcal{O}(k \cdot 2^k)$ – once we replace 2 with 5 since rLTL is 5-valued while LTL is 2-valued.

Similarly to LTL, our translation relies on so-called expansion rules, which we introduce in Section 4.1. Based on these rules, we present the translation from rLTL($\boxdot, \diamondsuit$) to generalized Büchi automata in Section 4.2. Subsequently, we consider model checking in Section 4.3 and reactive synthesis in Section 4.4.

4.1 Expansion Rules

The temporal operators $\boxdot$ and $\diamondsuit$ have expansion rules, which relate valuations at time ℓ to that of time $\ell + 1$, similar to their LTL counterparts $\Box$ and $\Diamond$ (see Baier and Katoen [3] for an in-depth discussion of LTL expansion rules). The following proposition states these rules.

▶ **Proposition 6** (Expansion Rules). *For any rLTL($\boxdot, \diamondsuit$) formula φ, any $\sigma \in \Sigma^\omega$, any $\ell \in \mathbb{N}$, and any valuation V, the following equalities (called* expansion rules*) hold:*

$$V_1(\sigma_{\ell..}, \boxdot\varphi) = \min\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{19}$$
$$V_2(\sigma_{\ell..}, \boxdot\varphi) = \max\{V_1(\sigma_{\ell..}, \boxdot\varphi), V_2(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{20}$$
$$V_3(\sigma_{\ell..}, \boxdot\varphi) = \min\{V_4(\sigma_{\ell..}, \boxdot\varphi), V_3(\sigma_{\ell+1..}, \boxdot\varphi)\}, \tag{21}$$
$$V_4(\sigma_{\ell..}, \boxdot\varphi) = \max\{V_4(\sigma_{\ell..}, \varphi), V_4(\sigma_{\ell+1..}, \boxdot\varphi)\}. \tag{22}$$

Moreover, the following holds for each $k \in \{1, \dots, 4\}$:

$$V_k(\sigma_{\ell..}, \diamondsuit\varphi) = \max\{V_k(\sigma_{\ell..}, \varphi), V_k(\sigma_{\ell+1..}, \diamondsuit\varphi)\}. \tag{23}$$

Before we prove this proposition, let us highlight that Equation (20) does not only recur on V_2 but also on V_1 (an analogous observation is true for Equation (21)). In fact, by recurring on $V_1(\sigma_{\ell..}, \boxdot\varphi)$ instead of $\sup_{k \geq \ell} V_2(\sigma_{k..}, \varphi)$, as one might have expected, we avoid the intermediate computation of $\sup_{k \geq \ell} V_2(\sigma_{k..}, \varphi)$ by the generalized Büchi automaton and, thereby, save auxiliary memory. This is the key property that allows us to prevent an unduly growth in the size of the resulting Büchi automaton and to achieve the desired bound on the number of states.

Proof of Proposition 6. Equality (19) follows directly from the properties of inf:

$$\begin{aligned} V_1(\sigma_{\ell..}, \boxdot\varphi) &= \inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) \\ &= \inf\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \varphi), V_1(\sigma_{\ell+2..}, \varphi), \ldots\} \\ &= \inf\{V_1(\sigma_{\ell..}, \varphi), \inf\{V_1(\sigma_{\ell+1..}, \varphi), V_1(\sigma_{\ell+2..}, \varphi), \ldots\}\} \\ &= \min\left\{V_1(\sigma_{\ell..}, \varphi), \inf_{i\geq\ell+1} V_1(\sigma_{i..}, \varphi)\right\} \\ &= \min\{V_1(\sigma_{\ell..}, \varphi), V_1(\sigma_{\ell+1..}, \boxdot\varphi)\}. \end{aligned}$$

A similar argument using the properties of sup shows that

$$V_2(\sigma_{\ell..}, \boxdot\varphi) = \sup_{j\geq\ell}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\}.$$

To conclude the proof of Equality (20), we need to replace the term $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi)$ inside the max by $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi)$; in other words, we must prove the last equality in the equation

$$\begin{aligned} \max\{V_1(\sigma_{\ell..}, \boxdot\varphi), V_2(\sigma_{\ell+1..}, \boxdot\varphi)\} &= \max\left\{\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} \\ &= \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} \end{aligned} \tag{24}$$

holds for every sequence $\sigma \in \Sigma^\omega$, every rLTL($\boxdot, \diamondsuit$) formula φ, and any valuation V.

To this end, we consider two separate cases. The first case is $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 1$ and immediately leads to the desired equality

$$\begin{aligned} \max\left\{\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\} &= 1 \\ &= \max\left\{\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi), \sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi)\right\}. \end{aligned}$$

The second case is $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 0$ and the desired equality reduces to

$$\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = \inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi).$$

We now note that $\sup_{j\geq\ell+1}\inf_{i\geq j} V_2(\sigma_{i..}, \varphi) = 0$ implies $\inf_{i\geq\ell+1} V_2(\sigma_{i..}, \varphi) = 0$, which implies $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi) = 0$. To conclude the proof, we must show $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = 0$. We recall that every element $b = (b_1, b_2, b_3, b_4) \in \mathbb{B}_4$ satisfies $b_1 \leq b_2$. In particular, we have $V_1(\sigma_{i..}, \varphi) \leq V_2(\sigma_{i..}, \varphi)$ for every $i \in \mathbb{N}$, and it follows from the monotonicity properties of inf that

$$\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) \leq \inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi).$$

The proof of Equality (20) is now finished by noting that the previous inequality and $\inf_{i\geq\ell} V_2(\sigma_{i..}, \varphi) = 0$ imply $\inf_{i\geq\ell} V_1(\sigma_{i..}, \varphi) = 0$.

The proof of Equality (21) is dual to the proof of Equality (20), while the proof of Equality (22) is dual to the proof of Equality (19). ◀

4.2 From rLTL($\boxdot, \diamondsuit$) to Generalized Büchi Automata

Let us begin this section by briefly recalling the definition of generalized Büchi automata.

▶ **Definition 7** (Generalized Büchi automaton)**.** A *generalized Büchi automaton (GBA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ consisting of a nonempty, finite set Q of states, a (finite) input alphabet Σ, an initial state $q_0 \in Q$, a (nondeterministic) transition relation $\Delta \in Q \times \Sigma \times Q$, and a set $\mathcal{F} \subseteq 2^Q$ denoting the acceptance conditions.

The *run* of a generalized Büchi automaton on a word $\sigma \in \Sigma^\omega$ (also called *input*) is an infinite sequence of states $\rho = q_0 q_1 \ldots \in Q^\omega$ satisfying $(q_i, \sigma(i), q_{i+1}) \in \Delta$ for all $i \in \mathbb{N}$ (note that every run starts in the initial state q_0). Given a run $\rho = q_0 q_1 \ldots$, we denote the set of states occurring infinitely often during ρ by $\mathrm{Inf}(\rho) = \{q \in Q \mid \forall i \in \mathbb{N}\ \exists j \geq i \colon q_j = q\}$. A run ρ is called *accepting* if $\mathrm{Inf}(\rho) \cap F \neq \emptyset$ for all $F \in \mathcal{F}$ (i.e., the run visits a state of each set $F \in \mathcal{F}$ infinitely often). The *language* of a generalized Büchi automaton $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of all infinite words $\sigma \in \Sigma^\omega$ for which an accepting run of $\mathcal{A}$ exists.

Having recapitulated these basic definitions, we can now turn to our translation from rLTL($\boxdot, \diamondsuit$) to generalized Büchi automata. This translation is an adaptation of a standard translation from LTL to generalized Büchi automata that constructs an automaton whose states correspond to valuations of the given LTL formula. An important concept in this context is that of the so-called φ-*expansion*: given an rLTL($\boxdot, \diamondsuit$) formula φ, the φ-expansion of an infinite word $\sigma \in \Sigma^\omega$ tracks the evaluation of φ and its subformulas at each position of σ. The key idea then is to construct a generalized Büchi automaton $\mathcal{A}_\varphi$ that nondeterministically guesses the φ-expansion step-by-step when reading its input and verifies the guess by means of its acceptance conditions. A formal definition of the φ-expansion is given below. To ease the following presentation, we denote the set of sub-formulas of an rLTL($\boxdot, \diamondsuit$) formula φ, which is defined in the usual way, by *closure of* φ, abbreviated as $\mathrm{cl}(\varphi)$.

▶ **Definition 8** (φ-expansion)**.** Let φ be an rLTL($\boxdot, \diamondsuit$) formula. The φ-*expansion* of an infinite word $\sigma \in \Sigma^\omega$ is a mapping $\eta \colon cl(\varphi) \times \mathbb{N} \to \mathbb{B}_4$ satisfying $\eta(\psi, i) = V(\sigma_{i..}, \psi)$ for all $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$.

Note that the φ-expansion is unique for a given word and subsumes the valuation of φ in the sense that $V(\varphi, \sigma) = \eta(\varphi, 0)$. Although the definition of the φ-expansion is not constructive, we can introduce constraints that completely characterize the φ-expansion of a given word. The pivotal idea is to impose constraints for local consistency (e.g., $\eta(\neg\psi, i)$ for $\psi \in \mathrm{cl}(\varphi)$ at some position $i \in \mathbb{N}$ has to be $\overline{\eta(\psi, i)}$) and to exploit the expansion rules of Proposition 6 to relate $\eta(\psi, i)$ and $\eta(\psi, i+1)$. As in the case of valuations V, we use the shorthand-notation $\eta_j(\psi, i)$ instead of the more verbose expression $\pi_j(\eta(\psi, i))$.

In the following, let $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$. The first type of constraints (*local constraints*) are as follows:

A1. If $\psi = p$, then $\eta(\psi, i) = \begin{cases} 0000 & \text{if } p \notin \sigma(i); \text{ and} \\ 1111 & \text{if } p \in \sigma(i). \end{cases}$

A2. If $\psi = \neg\psi_1$, then $\eta(\psi, i) = \overline{\eta(\psi_1, i)}$.

A3. If $\psi = \psi_1 \wedge \psi_2$, then $\eta(\psi, i) = \min\{\eta(\psi_1, i), \eta(\psi_2, i)\}$.

A4. If $\psi = \psi_1 \vee \psi_2$, then $\eta(\psi, i) = \max\{\eta(\psi_1, i), \eta(\psi_2, i)\}$.

A5. If $\psi = \psi_1 \Rightarrow \psi_2$, then $\eta(\psi, i) = \eta(\psi_1, i) \to \eta(\psi_2, i)$.

A6. If $\psi = \diamondsuit\psi_1$, then $\eta(\psi, i) = (b_1, b_2, b_3, b_4)$ where $b_j = \max\{\eta_j(\psi_1, i), \eta_j(\psi, i+1)\}$ for $j \in \{1, \ldots, 4\}$.

A7. If $\psi = \boxdot\psi_1$, then $\eta(\psi, i) = (b_1, b_2, b_3, b4)$ where

(a) $b_1 = \min\{\eta_1(\psi_1, i), \eta_1(\psi, i+1)\}$,
(b) $b_2 = \max\{b_1, \eta_2(\psi, i+1)\}$,
(c) $b_3 = \min\{b_4, \eta_3(\psi, i+1)\}$, and
(d) $b_4 = \max\{\eta_4(\psi_1, i), \eta_4(\psi, i+1)\}$.

To ensure satisfaction of the subformulas involving the temporal operators $\diamondsuit$ and $\boxdot$, we add the following further constraints (*non-local constraints*). These constraints are derived from the expansion rules and translate directly into Büchi conditions.

B1. For each $\diamondsuit\psi \in cl(\varphi)$ and $j \in \{1, \dots, 4\}$, there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_j(\diamondsuit\psi, \ell) = 1$ and $\eta_j(\psi, \ell) = 0$.

B2. For each $\boxdot\psi \in cl(\varphi)$,
(a) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_1(\boxdot\psi, \ell) = 0$ and $\eta_1(\psi, \ell) = 1$;
(b) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_2(\boxdot\psi, \ell) = 1$ and $\eta_1(\boxdot\psi, \ell) = 0$;
(c) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_3(\boxdot\psi, \ell) = 0$ and $\eta_4(\boxdot\psi, \ell) = 1$;
(d) there exists no $k \in \mathbb{N}$ such that for every $\ell \geq k$ both $\eta_4(\boxdot\psi, \ell) = 1$ and $\eta_4(\psi, \ell) = 0$.

In fact, these constraints completely characterize the φ-expansion of a given word, which is formally proven in Appendix A.

▶ **Lemma 9.** *Given an rLTL($\boxdot, \diamondsuit$) formula φ over the atomic propositions $\mathcal{P}$ and an infinite word $\sigma \in \Sigma^\omega$ where $\Sigma = 2^\mathcal{P}$, let $\eta\colon cl(\varphi) \times \mathbb{N} \to \mathbb{B}_4$ be a mapping satisfying the compatibility constraints A1 to B2. Then, η is uniquely determined, and it is the φ-expansion of σ.*

Before we define the generalized Büchi automaton $\mathcal{A}_\varphi$ formally, let us sketch its construction. The states of $\mathcal{A}_\varphi$ are mappings $\mu\colon cl(\varphi) \to \mathbb{B}_4$, which encode the φ-expansion of a word σ in the sense that the sequence of states $\mu_0\mu_1\dots$ constituting an accepting run on σ satisfies $\mu_i(\psi) = \eta_i(\psi)$ for all $i \in \mathbb{N}$ and $\psi \in cl(\varphi)$. Clearly, the only states (i.e., mappings μ) of interest are those consistent with the local compatibility constraints A1 to A5.[5] Thus, in order to ease the following definition, we denote the set of such mappings by S (note that the cardinality of S is bounded by $|\mathbb{B}_4|^{|cl(\varphi)|} = 5^{|cl(\varphi)|}$).

When reading an input-word, the automaton $\mathcal{A}_\varphi$ uses its transitions to verify that its guess satisfies the local constraints and its acceptance condition to verify the non-local constraints. The latter is achieved by adding a Büchi condition for each of the Conditions B1 and B2, which translate the respective conditions in a straightforward manner. To capture the many-valued semantics of rLTL($\square, \diamondsuit$), we define the automaton without an initial state. Instead, we introduce states q_b for each $b \in \mathbb{B}_4$ with the property that $\mathcal{A}_\varphi$ accepts a word $\sigma \in \Sigma^\omega$ when starting in the state q_b if and only if $V(\sigma, \varphi) = b$. Thus, an accepting run starting in q_b signals that φ evaluates on σ to b. The formal definition of $\mathcal{A}_\varphi$ is as follows.

▶ **Definition 10** (Automaton $\mathcal{A}_\varphi$)**.** Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the atomic propositions $\mathcal{P}$. Additionally, let $\Sigma = 2^\mathcal{P}$, $a \in \Sigma$, and S be the set of functions $\mu\colon cl(\varphi) \to \mathbb{B}_4$ that satisfy Conditions A1 to A5. We define the *generalized Büchi automaton* $\mathcal{A}_\varphi = (Q, \Sigma, \Delta, \mathcal{F})$ as follows:

- $Q = \{q_b \mid b \in \mathbb{B}_4\} \cup S$;
- the transition relation is defined by:
  - $(q_b, a, \mu) \in \Delta$ if and only if $\mu(\varphi) = b$ and $\mu(p) = \begin{cases} 1111 & \text{if } p \in a \cap \mathrm{cl}(\varphi); \text{ and} \\ 0000 & \text{if } p \in \mathrm{cl}(\varphi) \setminus a; \end{cases}$

[5] By this we mean that the conditions are satisfied if we substitute μ for η.

- $(\mu, a, \mu') \in \Delta$ if and only if the pair $(\mu,\, \mu')$ satisfies Conditions A6 and A7 as well as
$$\mu'(p) = \begin{cases} 1111 & \text{if } p \in a \cap \mathrm{cl}(\varphi); \text{ and} \\ 0000 & \text{if } p \in \mathrm{cl}(\varphi) \setminus a; \end{cases}$$

- $\mathcal{F}$ is the union of the following sets:
  - for each $\diamondsuit\psi \in cl(\varphi)$, we introduce for each $j \in \{1, \ldots, 4\}$ the set
$$F_{\diamondsuit\psi,j} = \{\mu \in S \mid \pi_j(\mu(\diamondsuit\psi)) = 0 \text{ or } \pi_j(\mu(\psi)) = 1\};$$
  - for each $\boxdot\psi \in cl(\varphi)$, we introduce the sets:
$$\begin{aligned} F_{\boxdot\psi,1} &= \{\mu \in S \mid \pi_1(\mu(\boxdot\psi)) = 1 \text{ or } \pi_1(\mu(\psi)) = 0\} \\ F_{\boxdot\psi,2} &= \{\mu \in S \mid \pi_2(\mu(\boxdot\psi)) = 0 \text{ or } \pi_1(\mu(\boxdot\psi)) = 1\} \\ F_{\boxdot\psi,3} &= \{\mu \in S \mid \pi_3(\mu(\boxdot\psi)) = 1 \text{ or } \pi_4(\mu(\boxdot\psi)) = 0\} \\ F_{\boxdot\psi,4} &= \{\mu \in S \mid \pi_4(\mu(\boxdot\psi)) = 0 \text{ or } \pi_4(\mu(\psi)) = 1\} \end{aligned}$$

Definition 10 ensures that $\mathcal{A}_\varphi$ accepts $\sigma \in \Sigma^\omega$ if and only if there exists a run $q_b, \mu_0, \mu_1, \ldots$ on σ that visits each $F \in \mathcal{F}$ infinitely often. As an example, suppose that a run visits the set $F_{\diamondsuit\psi,1}$ for $\diamondsuit\psi \in cl(\varphi)$ infinitely often (i.e., $\pi_1(\mu_i(\diamondsuit\psi)) = 0$ or $\pi_1(\mu_i(\psi)) = 1$ holds for infinitely many $i \in \mathbb{N}$). This means that it never happens that from some $k \in \mathbb{N}$ onward both $\pi_1(\mu_k(\diamondsuit\psi)) = 1$ and $\pi_1(\mu_k(\psi)) = 0$. Hence, Condition B1 is fulfilled. Similarly, the remaining sets $F \in \mathcal{F}$ make sure that Conditions B1 and B2 are indeed satisfied. Moreover, the definition of Δ ensures that Conditions A1 to A7 are satisfied along an accepting run of $\mathcal{A}_\varphi$ on σ and, therefore, this run in fact forms the φ-expansion of σ (and is unique). Finally, by using different initial states, we make sure that $\mathcal{A}_\varphi$ accepts σ starting from q_b if only if $b = V(\sigma, \varphi)$ (since all outgoing transitions lead to states μ with $\mu(\varphi) = b$). As a consequence, we obtain the following result.

▶ **Theorem 11.** *Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the set $\mathcal{P}$ of atomic propositions, $\Sigma = 2^{\mathcal{P}}$, and $b \in \mathbb{B}_4$. Then, $\mathcal{A}_\varphi$ accepts $\sigma \in \Sigma^\omega$ when starting in state q_b if and only if $V(\sigma, \varphi) = b$.*

For notational convenience, we denote the generalized Büchi automaton $\mathcal{A}_\varphi$ with initial state q_b by $\mathcal{A}^b_\varphi$. In addition, given a set $B \subseteq \mathbb{B}_4$, we denote the generalized Büchi automaton accepting the union $\bigcup_{b \in B} L(\mathcal{A}^b_\varphi)$ by $\mathcal{A}^B_\varphi$. The construction of $\mathcal{A}^B_\varphi$ is straightforward: first, we add a new state, say q_0, to $\mathcal{A}_\varphi$ and designate it as initial state; second, we add ε-transitions (q_0, ε, q_b) for each $b \in B$, which can subsequently be removed in the same manner as for finite automata with ε-transitions.

We finish this section with a remark about the size of the automata $\mathcal{A}_\varphi$ and $\mathcal{A}^B_\varphi$. Recalls that $\mathcal{A}_\varphi$'s state are (mainly) functions $\mu\colon cl(\varphi) \to \mathbb{B}_4$, of which there are $|\mathbb{B}_4|^{|\mathrm{cl}(\varphi)|} = 5^{|\mathrm{cl}(\varphi)|}$ many. Moreover, recall that we use four acceptance conditions per $\boxdot$ and $\diamondsuit$ operator.

▶ Remark. Both the automaton $\mathcal{A}_\varphi$ and $\mathcal{A}^B_\varphi$ have $\mathcal{O}(5^{|\mathrm{cl}(\varphi)|})$ states and at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

4.3 Model Checking

Broadly speaking, the model checking problem asks whether the model of a given system exhibits a specified behavior (which is described as an rLTL($\boxdot, \diamondsuit$) formula in our case). Usually, a system is modeled as a Kripke structure, which is, for the sake of model checking, translated into a Büchi automaton whose language corresponds to the unraveling of the Kripke structure. For reasons of simplicity, we consider a system – more precisely, a model

thereof – already to be given as a (generalized) Büchi automaton. This leads to the following formulation of the model checking problem.

▶ **Problem 1** (Model checking). *Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the set $\mathcal{P}$ of atomic propositions, let $\mathcal{A}$ be a generalized Büchi automaton over the alphabet $2^{\mathcal{P}}$, and let $B \subseteq \mathbb{B}_4$. Does $V(\sigma, \varphi) \in B$ hold for all $\sigma \in L(\mathcal{A})$?*

Our translation of rLTL($\boxdot, \diamondsuit$) formulas into a generalized Büchi automaton provides a straightforward means to answer the model checking problem: one simply constructs $\mathcal{A}_\varphi^B$ and checks $L(\mathcal{A}) \subseteq L(\mathcal{A}_\varphi^B)$. However, the naive attempt to check this inclusion (i.e., checking whether $L(\mathcal{A}) \cap (\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B)) = \emptyset$ holds) would require to complement $\mathcal{A}_\varphi^B$, which we clearly want to avoid due to the inevitable exponential blowup; moreover, the equality $\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B) = L(\mathcal{A}_{\neg\varphi}^B)$ does not hold in general. Instead, we exploit the fact that one can write the complement of $L(\mathcal{A}_\varphi^B)$ as $\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B) = L(\mathcal{A}_\varphi^{B'})$ for $B' = \mathbb{B}_4 \setminus B$, which leads to the following result; a proof can be found in Appendix B.

▶ **Theorem 12.** *One can decide the model checking problem (Problem 1) for the Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ and the rLTL($\boxdot, \diamondsuit$) formula φ in time $\mathcal{O}\big((|\mathcal{F}| + |\mathrm{cl}(\varphi)|) \cdot |Q| \cdot 5^{|\mathrm{cl}(\varphi)|}\big)$.*

It is worth mentioning that the corresponding optimization problem, namely to find the largest $b \in \mathbb{B}_4$ such that $V(\sigma, \varphi) \geq b$ holds for all $\sigma \in L(\mathcal{A})$, can be solved by repeatedly solving Problem 1 for decreasing values of b and, hence, falls into the complexity same class.

4.4 Synthesis of Reactive Systems

In the context of reactive synthesis, we consider infinite-duration two-player games over finite graphs with rLTL($\boxdot, \diamondsuit$) winning conditions. In particular, we show, given a game with rLTL($\boxdot, \diamondsuit$) winning condition, how to construct a finite-state winning strategy. Throughout this section, we assume familiarity with games over finite graphs and follow the definitions and notations by Grädel, Thomas, and Wilke [11].

We consider games of the following kind.

▶ **Definition 13** (rLTL($\boxdot, \diamondsuit$) games). Let $\mathcal{P}$ be a finite set of atomic propositions. An *rLTL($\boxdot, \diamondsuit$) game* is a pair $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ consisting of

- a finite, labeled *game graph* $\mathcal{G} = (V, E, \lambda)$ where V is a finite set of vertices that is partitioned into two disjoint sets $V_0, V_1 \subseteq V$, $E \subseteq V \times V$ is an edge relation, and $\lambda \colon V \to 2^{\mathcal{P}}$ is a function labeling each vertex with atomic propositions; and
- a pair (φ, B) consisting of an rLTL($\boxdot, \diamondsuit$) formula φ over $\mathcal{P}$ and a set $B \subseteq \mathbb{B}_4$ (this pair constitutes the *winning condition* as we formalize shortly).

An rLTL($\boxdot, \diamondsuit$) game is played as usual by two players, Player 0 and Player 1, who construct a *play* $\rho = v_0 v_1 \ldots \in V^\omega$ (i.e., an infinite sequence of vertices) by moving a token along the edges of the game graph. A play $\rho = v_0 v_1 \ldots$ induces an infinite word $\lambda(\rho) = \lambda(v_0)\lambda(v_1) \ldots \in (2^{\mathcal{P}})^\omega$, and the value of the formula φ on $\lambda(\rho)$ determines the winner of the play. More precisely, we call a play $\rho \in V^\omega$ *winning for Player 0* if $V(\lambda(\rho), \varphi) \in B$; symmetrically, we call a play *winning for Player 1* if it is *not winning* for Player 0. Finite-state strategies and winning strategies are defined as usual (see Grädel, Thomas, and Wilke [11] for further details).

Given an rLTL($\boxdot, \diamondsuit$) game and a vertex $v \in V$, we are interested in *solving the game* (i.e., in deciding which player has a winning strategy from v and in computing such a strategy), which is formalized next.

▶ **Problem 2** (Solving rLTL($\boxdot, \diamondsuit$) games). *Let an rLTL($\boxdot, \diamondsuit$) game $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ over the set V of vertices and a vertex $v_0 \in V$ be given.*

1. *Determine the player who has a winning strategy from vertex v_0.*
2. *Compute a winning strategy from vertex v_0.*

To solve this problem, we follow the Safra-based approach using the following four-step process. A detailed description of each step can be found in Appendix C.

1. We construct a nondeterministic Büchi automaton $\mathcal{B}^B_\varphi$ with $L(\mathcal{B}^B_\varphi) = \{\sigma \in (2^{\mathcal{P}})^\omega \mid V(\sigma, \varphi) \in B\}$.
2. We determinize $\mathcal{B}^B_\varphi$ using Safra's construction [20], resulting in a (deterministic) Rabin automaton $\mathcal{C}^B_\varphi$ that is language-equivalent to $\mathcal{B}^B_\varphi$.
3. We construct a Rabin game $\mathfrak{G}'$ by taking the product of the game graph $\mathcal{G}$ and the Rabin automaton $\mathcal{C}^B_\varphi$.
4. We apply standard techniques to solve $\mathfrak{G}'$, which allows us to decide which player has a winning strategy from v and to construct a winning strategy for the corresponding player.

Using this process, we obtain the following results (the complexity results immediately follow from the size of $\mathcal{A}_\varphi$, the subsequent Safra construction, and the standard technique to solve the resulting Rabin game).

▶ **Theorem 14.** *Given an rLTL($\boxdot, \diamondsuit$) game $\mathfrak{G} = (\mathcal{G}, (\varphi, B))$ with $\mathcal{G} = (V, E, \lambda)$ and a vertex $v_0 \in V$, one can*

1. *decide which player has a winning strategy from v_0 (i.e., Problem 2 Part 1) and*
2. *compute a winning strategy for the corresponding player (i.e., Problem 2 Part 2)*

in time $\mathcal{O}(n^{k+3}kk!)$ where $n = |V| \cdot 2^{5^{c_0|\mathrm{cl}(\varphi)|}}$, $k = 5^{c_1|\mathrm{cl}(\varphi)|}$, and c_0, c_1 are suitable constants.

5 Quality is Dual to Robustness

We motivated rLTL($\boxdot, \diamondsuit$) by the need to distinguish between the different ways in which safety properties can be violated. One can take a dual view and seek to distinguish between the different ways in which guarantee properties are satisfied. To illustrate this point, consider the LTL formula $\diamondsuit p \Rightarrow \diamondsuit q$ where $\diamondsuit p$ is an environment assumption and $\diamondsuit q$ is a system guarantee. According to the motto *more is better* we would prefer the system to guarantee the stronger property $\square\diamondsuit q$ whenever the environment satisfies the stronger property $\square\diamondsuit p$. By now, the reader can already complete our argument: $\diamondsuit\square p$ should lead to $\diamondsuit\square q$ and $\square p$ should lead to $\square q$. Formalizing these ideas would still take us to a 5-valued logic where, however, negation needs to be defined differently. Although we can still use the linear order

$$0000 \prec 0001 \prec 0011 \prec 0111 \prec 1111$$

on the set of truth values, one now needs to interpret the values differently. The value 0000 still corresponds to *false* but the remaining truth values now correspond to different quality values for *true* with 0001 being the lowest quality and 1111 the highest. Negation should then take 0000 to 1111 and all the remaining truth values to 0000. Such negation is no more than the intuitionistic negation already discussed in Section 3.3 and would equip $\mathbb{B}_4$ with the structure of an Heyting algebra instead of the da Costa algebra used in this paper. This observation justifies the title of this section and suggests the following question: is there an extension of LTL that can be used to reason about *both* robustness and quality? This is a question we leave for further research.

6 Discussion

The logic rLTL offers a transparent way to reason about the robustness of LTL specifications. Given an LTL formula φ, one obtains the corresponding rLTL formula ψ simply by dotting the temporal operators in φ. The technical development of the semantics was based on the insight that the temporal operators $\Box$ and $\Diamond$ count how often the formula they are applied to is satisfied thereby leading to a 5-valued logic. We studied the verification and synthesis problems for rLTL and showed they can be solved in exponential and doubly exponential time, respectively. These complexity bounds are the same as those for LTL once we replace 2, since LTL is Boolean valued, with 5, since rLTL is 5-valued. It remains an open problem to determine if these complexity upper bounds are tight. In addition to this question, we sketched in Section 5 a variant of rLTL tailored to quality and raised the question of how to combine robustness and quality in a single logic.

References

1 Shaull Almagor and Orna Kupferman. Latticed-ltl synthesis in the presence of noisy inputs. In *FOSSACS 2014*, volume 8412 of *LNCS*, pages 226–241. Springer, 2014.

2 Rajeev Alur, Aditya Kanade, and Gera Weiss. Ranking automata and games for prioritized requirements. In *CAV 2008*, volume 5123 of *LNCS*, pages 240–253. Springer, 2008.

3 Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

4 Roderick Bloem, Krishnendu Chatterjee, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Robustness in the presence of liveness. In *CAV 2010*, volume 6174 of *LNCS*, pages 410–424. Springer, 2010.

5 Roderick Bloem, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Synthesizing robust systems. In *FMCAD 2009*, pages 85–92. IEEE, 2009.

6 Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. Continuity analysis of programs. In *POPL 2010*, pages 57–70. ACM, 2010.

7 Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS 2010*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.

8 Laurent Doyen, Thomas A. Henzinger, Axel Legay, and Dejan Nickovic. Robustness of sequential circuits. In *ACSD 2010*, pages 77–84. IEEE, 2010.

9 Rüdiger Ehlers and Ufuk Topcu. Resilience to intermittent assumption violations in reactive synthesis. In *HSCC 2014*, pages 203–212. ACM, 2014.

10 Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

11 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

12 B. Jonsson and A. Tarski. Boolean algebras with operators. Part I. *American Journal of Mathematics*, 73(4):891–939, 1951.

13 Orna Kupferman and Yoad Lustig. Lattice automata. In *VMCAI 2007*, volume 4349 of *LNCS*, pages 199–213. Springer, 2007.

14 Rupak Majumdar and Indranil Saha. Symbolic robustness analysis. In *RTSS 2009*, pages 355–363. IEEE, 2009.

15 Doron Peled and Thomas Wilke. Stutter-invariant temporal properties are expressible without the next-time operator. *Information Processing Letters*, 63(5):243–246, 1997.

16 Dominique Perrin and Jean-Eric Pin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004.

17 Nir Piterman and Amir Pnueli. Faster solutions of rabin and streett games. In *LICS 2006*, pages 275–284. IEEE, 2006.

18 G. Priest. Dualising Intuitionist Logic. *Principia*, 13(2):165–184, 2009.

19 Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinformatics*, 25(12), 2009. doi:10.1093/bioinformatics/btp200.

20 Shmuel Safra. On the complexity of omega-automata. In *FOCS 1988*, pages 319–327. IEEE, 1988.

21 P. Tabuada and D. Neider. Robust Linear Temporal Logic, 2015. arXiv:1510.08970.

22 Paulo Tabuada, Sina Yamac Caliskan, Matthias Rungger, and Rupak Majumdar. Towards robustness for cyber-physical systems. *IEEE Trans. Automat. Contr.*, 59(12):3151–3163, 2014.

23 Milan Češka, David Šafránek, Sven Dražan, and Luboš Brim. Robustness analysis of stochastic biochemical systems. *PLoS ONE*, 9(4), 2014. doi:10.1371/journal.pone.0094553.

A φ-Expansion

Proof of Lemma 9. To prove the lemma, we need to establish that $V(\sigma_{i..}, \psi) = \eta(\psi, i)$ holds for all $\psi \in cl(\varphi)$ and $i \in \mathbb{N}$. The proof proceeds by structural induction over the subformulas of φ.

Base case. In the case of atomic propositions, the claim holds by definition of V.

Induction step. In the case of the operators $\neg$, $\vee$, $\wedge$, and $\Rightarrow$, the claim follows immediately from applying the induction hypothesis and by definition of V.

In the case of $\psi = \diamondsuit\psi_1$, a straightforward induction that applies (a) Condition A6, (b) the expansion rule for $\diamondsuit$ (see Proposition 6, Equation (22)), and (c) the induction hypothesis for ψ_1 (i.e., $V(\sigma_{i..}, \psi_1) = \eta(\psi_1, i)$ for all $i \in \mathbb{N}$) shows that the following is true for each $j \in \{1, \ldots, 4\}$: if $\eta_j(\psi_1, k) = 1$ for a $k \in \mathbb{N}$, then $\eta_j(\psi, \ell) = 1$ and, hence, $V_j(\sigma_{\ell..}, \psi) = \eta_j(\psi, \ell)$ for all $\ell \leq k$. Therefore, if infinitely many k with $\eta_j(\psi_1, k) = 1$ exist, then $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ for all $i \in \mathbb{N}$. If this is not the case, then there exists a $k \in \mathbb{N}$ such that $\eta_j(\psi_1, \ell) = 0$ for all $\ell \geq k$. Then, Condition B1 asserts for all $\ell \geq k$ that $\eta_j(\psi, \ell) = 0$ and, hence, $V_j(\sigma_{\ell..}, \psi) = \eta_j(\psi, \ell)$ is satisfied by the semantics of $\diamondsuit$ and the induction hypothesis for ψ_1; this, in turn, implies $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ for all $i \in \mathbb{N}$. These arguments are true for all $j \in \{1, \ldots, 4\}$ and, therefore, $V(\sigma_{i..}, \psi) = \eta(\psi, i)$ holds for all $i \in \mathbb{N}$.

The case $\psi = \boxdot\psi_1$ can be proven using similar arguments as in the case of the $\diamondsuit$ operator, but the semantics of $\boxdot$ requires to split the proof into four parts and prove $V_j(\sigma_{i..}, \psi) = \eta_j(\psi, i)$ individually for each $j \in \{1, \ldots, 4\}$. So as not to clutter this proof too much, we provide a detailed proof for $j = 1$ and skip the remaining. However, it is important to note that the claim needs to be proven first for $j = 1$ and $j = 4$ since the proofs for $j = 2$ and $j = 3$ rely thereon (the expansion rules recur on $V_1(\sigma_{i..}, \psi)$ and $V_4(\sigma_{i..}, \psi)$, respectively).

To prove $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$, we first observe that $\eta_1(\psi_1, k) = 0$ for a $k \in \mathbb{N}$ implies $V_1(\sigma_{\ell..}, \psi) = \eta_1(\psi, \ell)$ for all $\ell \leq k$; analogous to the case of the operator $\diamondsuit$, an induction using Condition A7(a), the expansion rule for $\boxdot$ (see Proposition 6, Formula (19)), and the induction hypothesis for ψ_1 establishes this. Hence, if infinitely many k with $\eta_1(\psi_1, k) = 0$ exist, then $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$. If this is not the case, then there exists a $k \in \mathbb{N}$ such that $\eta_1(\psi_1, \ell) = 1$ for all $\ell \geq k$. Then, Condition B2(a) asserts for all $\ell \geq k$ that $\eta_1(\psi, \ell)) = 1$ and, hence, $V_1(\sigma_{\ell..}, \psi) = \eta_1(\psi, \ell)$ holds by the

semantics of $\boxdot$ and the induction hypothesis of ψ_1. This implies $V_1(\sigma_{i..}, \psi) = \eta_1(\psi, i)$ for all $i \in \mathbb{N}$.

As mentioned above, the case $j = 4$ and the subsequent cases $j = 2$ and $j = 3$ are analogous. ◀

B Model Checking

Proof of Theorem 12. Let φ be an rLTL($\boxdot, \diamondsuit$) formula over the atomic propositions $\mathcal{P}$, $\mathcal{A} = (Q, \Sigma, q_0, \Delta, \mathcal{F})$ a generalized Büchi automaton over the alphabet $2^{\mathcal{P}}$, and $B \subseteq \mathbb{B}_4$.

First, let $B' = \mathbb{B}_4 \setminus B$. Then, it is not hard to verify that

$$\begin{aligned} V(\sigma, \varphi) \in B \text{ for all } \sigma \in L(\mathcal{A}) &\Leftrightarrow L(\mathcal{A}) \subseteq L(\mathcal{A}_\varphi^B) \\ &\Leftrightarrow L(\mathcal{A}) \cap \left(\Sigma^\omega \setminus L(\mathcal{A}_\varphi^B)\right) = \emptyset \\ &\Leftrightarrow L(\mathcal{A}) \cap \mathcal{A}_\varphi^{B'} = \emptyset. \end{aligned}$$

Recall that $\mathcal{A}_\varphi^{B'}$ has $5^{|\mathrm{cl}(\varphi)|} + 5$ states and also at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

Second, given two generalized Büchi automata $\mathcal{A}_1 = (Q_1, \Sigma, q_0^1, \Delta_1, \mathcal{F}_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, q_0^2, \Delta_2, \mathcal{F}_2)$, it is well-known that one can construct a generalized Büchi automaton accepting $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ using a simple product construction (see, e.g., [16]). This construction results in an automaton with $|Q_1| \cdot |Q_2|$ states and $|\mathcal{F}_1| + |\mathcal{F}_2|$ acceptance sets. Since $\mathcal{A}_\varphi^{B'}$ consists of $5^{|\mathrm{cl}(\varphi)|} + 5$ states and has at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets, this implies that one can construct a generalized Büchi automaton $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{A}_\varphi^{B'})$ consisting of $|Q| \cdot (5^{|\mathrm{cl}\varphi|} + 5)$ states and at most $|\mathcal{F}| + 4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets.

Finally, it is left to check whether $L(\mathcal{B}) = \emptyset$. This problem is fundamental in LTL model checking, and there exist efficient algorithms that solve this problem in time linear in the product of the number of states of the input automaton and the number of its acceptance sets (see, e.g., [3]). Hence, one can solve Problem 1 in $\mathcal{O}\big((|\mathcal{F}| + |\mathrm{cl}(\varphi)|) \cdot |Q| \cdot 5^{|\mathrm{cl}(\varphi)|}\big)$ time. ◀

C Computing Strategies in rLTL($\boxdot, \diamondsuit$)-Games

We compute a winning strategy in an rLTL($\boxdot, \diamondsuit$)-game using a four-step process:

1. First, we construct the generalized Büchi automaton $\mathcal{A}_\varphi^B$; recall that this automaton comprises $5^{|\mathrm{cl}(\varphi)|} + 5$ states and at most $4 \cdot |\mathrm{cl}(\varphi)|$ acceptance sets. Subsequently, we construct a nondeterministic Büchi automaton $\mathcal{B}_\varphi^B$ accepting the same language; the standard conversion results in a Büchi automaton that comprises $\mathcal{O}(4 \cdot |\mathrm{cl}(\varphi)| \cdot (5^{|\mathrm{cl}(\varphi)|} + 5))$ states.
2. Using Safra's determinization procedure [20], we obtain a (deterministic) Rabin automaton[6] $\mathcal{C}_\varphi^B$ that is language-equivalent to $\mathcal{B}_\varphi^B$. The automaton $\mathcal{C}_\varphi^B$ has $2^{5^{c_0 \cdot |\mathrm{cl}(\varphi)|}}$ states and $5^{c_1 \cdot |\mathrm{cl}(\varphi)|}$ Rabin pairs where $c_0 > c_1$ are suitable constants.
3. Next, we construct a Rabin game[7] as follows. We first construct the (unlabeled) product game graph $\mathcal{G}' = (V', E')$ of the game graph $\mathcal{G} = (V, E, \lambda)$ and the Rabin automaton

[6] A Rabin automaton is a tuple $\mathcal{C} = (Q, \Sigma, q_0, \delta, \Omega)$ where Q, Σ, and q_0 are as in Büchi automata, $\delta \colon Q \times \Sigma \to Q$ is a (deterministic) transition function, and $\Omega \subseteq 2^Q \times 2^Q$ is the acceptance condition. The *run* of a Rabin automaton on a word $\sigma \in \Sigma^\omega$ is an infinite sequence of states $\rho = q_0 q_1 \ldots$ satisfying $\delta(q_i, \sigma(i)) = q_{i+1}$ for all $i \in \mathbb{N}$. A run ρ is called *accepting* if there exists a pair $(E, F) \in \Omega$ such that $E \cap \mathrm{Inf}(\rho) = \emptyset$ and $F \cap \mathrm{Inf}(\rho) \neq \emptyset$.

[7] A Rabin game is a game played over an unlabeled game graph $\mathcal{G} = (V, E)$ with nonempty, finite set V of vertices and directed edge relation $E \subseteq V \times V$. The winning condition of a Rabin game is a set $\Omega \subseteq 2^V \times 2^V$, and a play $\rho = v_0 v_1 \ldots \in V^\omega$ is said to be winning for Player 0 if there exists a

$\mathcal{C}^B_\varphi = (Q, 2^{\mathcal{P}}, q_0, \delta, \Omega)$ such that $V' = V \times Q$ and

$$\big((v,q),(v',q')\big) \in E' \Leftrightarrow (v,v') \in E \text{ and } \delta(q,\lambda(v)) = q'.$$

Then, we define the Rabin winning condition of $\mathfrak{G}'$ to be

$$\Omega' = \big\{((V,E),(V,F)) \in V' \times V' \mid (E,F) \in \Omega\big\}.$$

The desired Rabin game is then $\mathfrak{G}' = (\mathcal{G}', \Omega')$.
An induction over the length of plays in $\mathfrak{G}'$ shows that Player 0 wins a play $\rho' = (v_0,q_0)(v_1,q_1)\dots$ if and only if Player 0 wins the play $\rho = v_0 v_1 \dots$ in $\mathfrak{G}$.

4. Finally, by applying the method by Piterman and Pnueli [17], we solve the resulting Rabin game in time $\mathcal{O}(n^{k+3}kk!)$ where $n = |V| \cdot 2^{5^{c_0|\mathrm{cl}(\varphi)|}}$ is the number of vertices and $k = 5^{c_1|\mathrm{cl}(\varphi)|}$ is the number of Rabin pairs of $\mathfrak{G}'$.

pair $(E,F) \in \Omega$ such that $E \cap \mathrm{Inf}(\rho) = \emptyset$ and $F \cap \mathrm{Inf}(\rho) \neq \emptyset$; by slight abuse of notation, $\mathrm{Inf}(\rho)$ here corresponds to the set of all vertices occurring infinitely often in the play ρ.

Models of λ-Calculus and the Weak MSO Logic*

Paweł Parys[1] and Szymon Toruńczyk[2]

1 **University of Warsaw, Warsaw, Poland**
`parys@mimuw.edu.pl`
2 **University of Warsaw, Warsaw, Poland**
`szymtor@mimuw.edu.pl`

Abstract

We study the Weak MSO logic in relationship to infinitary λ-calculus. We show that for every formula φ of Weak MSO there exists a finitary model of infinitary λ-calculus recognizing the set of infinitary λ-terms whose Böhm tree satisfies φ. The model is effective, in the sense that for every λY-term we can effectively compute its value in the model. In particular, given a finite λY-term, one can decide whether the resulting Böhm tree satisfies a given formula of Weak MSO, which is a special case of the result of Ong [16], which concerns unrestricted MSO. The existence of effective models for Weak MSO and MSO was proved earlier by Salvati and Walukiewicz [19, 20] but our proof uses a different method, as it does not involve automata, but works directly with logics.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases typed λ-calculus, models, Weak MSO logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.11

1 Introduction

In this paper, we study the model-checking problem on infinite trees generated by finite terms. More precisely, we consider simply typed λY-calculus – an extension of simply typed λ-calculus by a fixpoint operator Y. Any term K of simply typed λY-calculus generates an infinite tree, called the Böhm tree $BT(K)$, which reflects the control flow of the program corresponding to the term. We illustrate this with an example borrowed from [20], depicted in Figure 1.

Trees generated by terms of simply typed λY-calculus can be used to faithfully represent the workflow of programs in a language with higher-order functions. Traditionally, higher-order recursion schemes are used for this purpose [6, 12, 16, 13]; this formalism is equivalent to simply typed λY-calculus [18], and the translation between them is rather straightforward. Collapsible pushdown systems [9] and ordered tree-pushdown systems [5] are other equivalent formalisms.

The model-checking problem for λY-calculus is the following: given a closed λY-term K of the ground type, and a formula φ talking about trees, decide whether $BT(K) \models \varphi$. In this paper, we consider this problem, where the formula φ is a formula of Weak MSO logic. This logic extends first-order logic by allowing to quantify existentially and universally over finite sets of vertices. For example, the formula $\exists_{\mathsf{fin}} X. \forall x. (x \in X \iff a(x))$ holds in a tree t iff t has finitely many nodes labeled with a.

* Work supported by the National Science Center (decision DEC-2012/07/D/ST6/02443).

© Paweł Parys and Szymon Toruńczyk;
licensed under Creative Commons License CC-BY

$$Factorial(x) \equiv \texttt{if x} = \texttt{0 then 1 else x} \cdot Factorial(\texttt{x} - \texttt{1})$$
$$Fct \equiv Y(\lambda f.(\lambda x.\texttt{if } (\texttt{isZero } x)\ 1\ (x * f(x-1))))$$

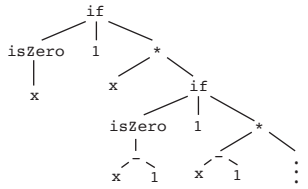


Figure 1 The factorial function, the corresponding λY-term, and its Böhm tree.

Weak MSO logic can capture many natural properties important in verification, such as safety or liveness. It is known from [16] that model-checking for a strictly larger logic is decidable, namely, for MSO logic, where quantification is not restricted to finite sets only. The same was later shown in [9, 13, 18, 23] using different methods.

The main result of our paper states that for every sentence φ of Weak MSO one can construct an effective and finitary *model* of simply typed λY-calculus, recognizing the set of trees in which φ is true. Roughly, this means that to every closed term K of λY-calculus we assign its value $[\![K]\!]_\varphi$, that can be effectively computed from K and φ. This assignment is such that only finitely many elements are used as values of terms of every fixed type. It also preserves composition, i.e., for the composition $K\,M$ of two terms, we have $[\![K\,M]\!]_\varphi = [\![K]\!]_\varphi\,[\![M]\!]_\varphi$, for appropriately defined composition $[\![K]\!]_\varphi\,[\![M]\!]_\varphi$ of elements of the model. Moreover, when K is of the ground type, we can determine whether $BT(K) \models \varphi$ holds seeing only $[\![K]\!]_\varphi$. A formal definition of a model is given in Section 2.

▶ **Theorem 1.1.** *For every sentence of φ of* Weak MSO *there exists an effective and finitary model of simply typed λY-calculus, that recognizes the set of trees in which φ is true.*

We remark that the term "model of λ-calculus" has various meanings in the literature [15, 2, 14], and the notion of model roughly described above is sometimes called an *applicative structure*. Our applicative structure has additional properties, namely that the value of a λY-term can be computed in a compositional way, from the values of its subterms.

The above result was proved earlier in [20]. In [19] and [7] analogous result was proved in full generality for MSO, and in [1] and [21] for properties definable by so-called TAC automata that are even less expressive than the Weak MSO logic. In this paper, we give yet another proof of this result for Weak MSO. Our approach is different than that of the cited papers, in that those papers work with and exploit the structure of parity automata corresponding to the logic, whereas we work directly with formulae.

It is argued in [21] that having a model has several other virtues in addition to providing decidability of the model-checking problem. In particular it allows to obtain the reflection property [4, 8] and the transfer theorem [17].

The paper [20], besides proving Theorem 1.1, provides a type system, such that typing a term using this system is equivalent to the Böhm tree being accepted by the fixed weak alternating automaton. The approach to model checking of Böhm trees using type systems has many advantages, and appears earlier in [13, 23, 11].

2 Preliminaries

Trees. Let Σ be a *ranked alphabet*, i.e., a set of symbols together with a rank function *rank* assigning a nonnegative integer to each of the symbols. Apart from Σ, we have a special symbol $\bot \notin \Sigma$ of rank 0, used to label a „missing part" of a tree. A Σ-*labeled* tree is a tree

which is rooted (there is a distinguished root node which is the ancestor of every node in the tree), node-labeled (every node has a label from $\Sigma \cup \{\bot\}$), ranked (a node with label of rank n has exactly n children), and ordered (the children of a node of rank n are numbered from 1 to n).

Weak MSO. The Weak MSO logic is a restriction of the MSO logic, in which set quantifiers range only over finite sets. For technical convenience, we use a variant of Weak MSO in which there are no first-order variables, and where set variables do not contain nodes labeled by $\bot$. It is easy to translate a formula from any standard syntax of Weak MSO to ours (at least when the alphabet Σ is finite). In the syntax of Weak MSO we have the following constructions:

$$\varphi ::= a(X) \mid X <_i Y \mid X \subseteq Y \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi' \mid \exists_{\mathsf{fin}} X.\varphi' \qquad \text{where } a \in \Sigma, i \in \mathbb{N}_+.$$

We evaluate formulae of Weak MSO in Σ-labeled trees. Set variables are interpreted as sets of nodes labeled by elements of Σ, and the semantics of formulae is defined as follows:

- $a(X)$ holds iff every node in X is labeled by a,
- $X <_i Y$ holds iff every node in Y is a (not necessarily proper) descendant of the i-th child of every node in X,
- $X \subseteq Y$, $\varphi_1 \wedge \varphi_2$, and $\neg\varphi'$ are defined as expected, and
- $\exists_{\mathsf{fin}} X.\varphi'$ holds iff φ' holds for some finite set X of nodes labeled by elements of Σ.

Infinitary λ-calculus. We consider infinitary, simply typed λ-calculus. In particular, each term has an associated type.

The set of *types* is constructed from a unique ground type o using a binary operation $\to$. Thus o is a type and if α, β are types, so is $(\alpha \to \beta)$. The order of a type is defined by: $ord(o) = 0$, and $ord(\alpha \to \beta) = \max(1 + ord(\alpha), ord(\beta))$. By convention, $\to$ associates to the right, i.e., $\alpha \to \beta \to \gamma$ is understood as $\alpha \to (\beta \to \gamma)$. Every type α can be uniquely written as $\alpha_1 \to \alpha_2 \to \ldots \to \alpha_n \to o$. The type $\underbrace{o \to \cdots \to o}_{k} \to o$ is denoted $o^k \to o$, where $o^0 \to o$ is simply o.

Infinitary λ-terms (or just *terms*) are defined by coinduction, according to the following rules:

- For each symbol $a \in \Sigma$ of rank r, $a^{o^r \to o}$ is a term.
- For each type α there are infinitely many variables $x^\alpha, y^\alpha, z^\alpha, \ldots$; each of them is a term.
- If $K^{\alpha\to\beta}$ and M^α are terms, then $(K^{\alpha\to\beta}\, M^\alpha)^\beta$ is a term.
- If K^β is a term and x^α is a variable, then $(\lambda x^\alpha.K^\beta)^{\alpha\to\beta}$ is a term.

We naturally identify two terms if they are α-equivalent. We often omit the type annotations of terms, but we keep in mind that every term has a fixed type. The set $FV(K)$ of free variables of a term K is defined as expected. A term K is *closed* if $FV(K) = \emptyset$. We denote terms by capital roman letters, e.g. $K, M, N, P, Q, \ldots$

λY-calculus. The syntax of λY-calculus is the same as of finite λ-calculus, extended by symbols $Y^{(\alpha\to\alpha)\to\alpha}$, for each type α. A term of λY-calculus is seen as a term of infinitary λ-calculus, by substituting each symbol $Y^{(\alpha\to\alpha)\to\alpha}$ by the unique term Y such that Y is equal to $\lambda M^{\alpha\to\alpha}.M\,(Y\,M)$. In this way, we view λY-calculus as a fragment of infinitary λ-calculus.

Trees Generated by Terms. Let K be a closed (infinitary) term of type o. The tree t *generated* by K is constructed by coinduction, as follows: if there is a sequence of β-reductions from K to a term of the form $a\,K_1\,\ldots\,K_r$, where a is a symbol (of rank r), then the root of the tree t has label a, and, for $i \in \{1,\ldots,r\}$, the i-th child of the root of t is the root of the tree generated by K_i. If there is no sequence of β-reductions from K to a term of the above form, the tree generated by K consists of a single node labeled by $\bot$.

The tree generated by K is also called the *Böhm tree* of K, and denoted $BT(K)$.

Models. In this paper, we choose the following definition of a model. An *applicative structure* D consists of a set $D[\alpha]$ for each type α, and of an application operation that to all elements $\sigma \in D[\alpha \to \beta]$ and $\tau \in D[\alpha]$ assigns an element $(\sigma\,\tau) \in D[\beta]$. A *$D$-valuation* is a partial function v that maps some variables of λ-calculus to elements of the applicative structure, so that $v(x^\alpha) \in D[\alpha]$.

A *model* $\mathcal{M}$ of infinitary λ-calculus consists of an applicative structure $D_\mathcal{M}$, called the *domain*, and a mapping that to each term K^α and to each $D_\mathcal{M}$-valuation v with $\mathrm{dom}(v) \supseteq FV(K)$ assigns an element $[\![K]\!]^v_\mathcal{M} \in D_\mathcal{M}[\alpha]$ so that

M1. $[\![K]\!]^v_\mathcal{M} = [\![K]\!]^w_\mathcal{M}$ whenever $v(x) = w(x)$ for all $x \in FV(K)$,
M2. $[\![K\,M]\!]^v_\mathcal{M} = [\![K]\!]^v_\mathcal{M}\,[\![M]\!]^v_\mathcal{M}$,
M3. $[\![\lambda x.K]\!]^v_\mathcal{M}\,\sigma = [\![K]\!]^{v[x\mapsto\sigma]}_\mathcal{M}$ (informally, $[\![\lambda x.K]\!]^v_\mathcal{M} = \lambda\sigma.[\![K]\!]^{v[x\mapsto\sigma]}_\mathcal{M}$), and
M4. $[\![K[M/x]]\!]^v_\mathcal{M} = [\![K]\!]^{v[x\mapsto[\![M]\!]^v_\mathcal{M}]}_\mathcal{M}$.

When K is closed, we write $[\![K]\!]_\mathcal{M}$ for $[\![K]\!]^v_\mathcal{M}$ (which is independent from v). The following fact follows easily from conditions M2–M4.

▶ **Fact 2.1.** *In every model $\mathcal{M}$ it holds that $[\![K]\!]^v_\mathcal{M} = [\![K']\!]^v_\mathcal{M}$ whenever K and K' are β-equivalent.*

We say that a model $\mathcal{M}$ *recognizes* a set of trees $\mathcal{L}$ if there is a subset F of $D_\mathcal{M}[o]$ such that for every closed term K^o, the tree generated by K belongs to $\mathcal{L}$ if and only if $[\![K]\!]_\mathcal{M} \in F$. We say that a model $\mathcal{M}$ is *finitary* if $D_\mathcal{M}[\alpha]$ is finite for each type α, and furthermore it is *effective*, if for a given λY-term K, one can compute $[\![K]\!]_\mathcal{M}$ (where K is treated as an infinitary λ-term obtained by expanding all Y symbols).

The main result of this paper, Theorem 1.1, states that there exists a finitary, effective model of infinitary λ-calculus recognizing any set of trees definable by a Weak MSO formula. Moreover, we will see that the value of a λY-term can be computed in a compositional way.

3 Phenotypes of Trees

In this section, we define phenotypes. These are objects that characterize properties of trees with respect to a given formula of Weak MSO, and that are compositional.

Let $\mathcal{F}$ be a finite set of variables of Weak MSO. An *$\mathcal{F}$-tree* is a pair $t \otimes v$, where t is a Σ-labeled tree, and v is a valuation of the variables in $\mathcal{F}$ in t. We write $\hat{t}$ to denote $\mathcal{F}$-trees, for some $\mathcal{F}$. If $\hat{t} = t \otimes v$ is a $\mathcal{F}$-tree and x is a node of $\hat{t}$, then the subtree of $\hat{t}$ rooted at x is the $\hat{F}$-tree $s \otimes w$ consisting of the subtree s of t rooted at t and the valuation w which is v restricted to the nodes of s (i.e., for a variable X, $w(X)$ is $v(X)$ intersected with the nodes of s).

Suppose that φ is a formula with free variables contained in $\mathcal{F}$ and $\hat{t} = t \otimes v$ is an $\mathcal{F}$-tree. Then we write $\hat{t} \models \varphi$ if $t, v \models \varphi$. We also define the *φ-phenotype* of $\hat{t} = t \otimes v$, denoted $[\hat{t}]_\varphi$, by induction on the size of the formula φ as follows:

- if φ is of the form $a(X)$ (for some symbol $a \in \Sigma$) or $X \subseteq Y$ then $[\hat{t}]_\varphi$ is the logical value of φ in $\hat{t}$, i.e., *true* if $t, v \models \varphi$ and *false* otherwise,
- if φ is of the form $X <_i Y$, then $[\hat{t}]_\varphi$ is the triple whose first element is the logical value of φ in $\hat{t}$, the second element is *true* if $v(X) = \emptyset$ and *false* otherwise, and the third element analogously for Y,
- if $\varphi = (\psi_1 \wedge \psi_2)$, then $[\hat{t}]_\varphi = ([\hat{t}]_{\psi_1}, [\hat{t}]_{\psi_2})$,
- if $\varphi = (\neg\psi)$, then $[\hat{t}]_\varphi = [\hat{t}]_\psi$, and
- if $\varphi = \exists_{\mathsf{fin}} X.\psi$, then $[\hat{t}]_\varphi = \{[t \otimes w]_\psi \mid w \text{ is a valuation extending } v \text{ to } X\}$.

For each φ, let Pht_φ denote the set of all potential φ-phenotypes. Namely, $\mathrm{Pht}_\varphi = \{\mathit{true}, \mathit{false}\}$ in the first case, $\mathrm{Pht}_\varphi = \{\mathit{true}, \mathit{false}\}^3$ in the second case, $\mathrm{Pht}_\varphi = \mathrm{Pht}_{\psi_1} \times \mathrm{Pht}_{\psi_2}$ in the third case, $\mathrm{Pht}_\varphi = \mathrm{Pht}_\psi$ in the fourth case, and $\mathrm{Pht}_\varphi = \mathcal{P}(\mathrm{Pht}_\psi)$ in the last case.

We are particularly interested in φ-phenotypes for valuations that map all set variables to the empty set. Thus for a tree t, by $[t]_\varphi$ we denote $[t \otimes v_\emptyset]_\varphi$, where $v_\emptyset$ is the valuation mapping all free variables of φ to the empty set.

We immediately see two facts. First, Pht_φ is finite for every φ. Second, the fact whether φ holds in $\hat{t}$ is determined by $[\hat{t}]_\varphi$.

Next, we observe that phenotypes of trees behave in a compositional way, as formalized below. For every symbol a and every formula φ, define a function $\mathrm{Comp}_{a,\varphi} \colon \mathcal{P}(\mathcal{V}) \times (\mathrm{Pht}_\varphi)^r \to \mathrm{Pht}_\varphi$, where r is the rank of a, and $\mathcal{V}$ is the set of variables that may occur in formulae of Weak MSO. The functions are defined so that the following lemma holds.

▶ **Lemma 3.1.** *Let φ be a formula with free variables contained in $\mathcal{F}$, and let $\hat{t} = t \otimes v$ be an $\mathcal{F}$-tree with root labeled by a symbol a of rank r. If $\mathcal{R}$ is the set those variables $X \in \mathcal{F}$ for which $v(X)$ contains the root of t, and $\hat{t}_i$ is the subtree of $\hat{t}$ rooted at the i-th child of the root (for $i \in \{1, \ldots, r\}$), then $[\hat{t}]_\varphi = \mathit{Comp}_{a,\varphi}(\mathcal{R}, [\hat{t}_1]_\varphi, \ldots, [\hat{t}_r]_\varphi)$.*

While defining $\mathrm{Comp}_{a,\varphi}$ we proceed by induction on the size of φ.

When φ is of the form $b(X)$ or $X \subseteq Y$, then we see that φ holds in $\hat{t}$ iff it holds in every subtree $\hat{t}_i$ and in the root of $\hat{t}$. Thus for $\varphi = b(X)$ as $\mathrm{Comp}_{a,\varphi}(\mathcal{R}, \tau_1, \ldots, \tau_r)$ we take *true* when $\tau_i = \mathit{true}$ for all $i \in \{1, \ldots, r\}$ and either $a = b$ or $X \notin \mathcal{R}$. For $\varphi = (X \subseteq Y)$ the last part of the condition is replaced by „if $X \in \mathcal{R}$ then $Y \in \mathcal{R}$".

Next, suppose that $\varphi = (X <_i Y)$. There are several ways in which X and Y can be distributed between the subtrees $\hat{t}_j$ and the root of $\hat{t}$ so that $X <_i Y$ holds in $\hat{t}$ (i.e. so that the first coordinate of $[t]_\varphi$ is *true*), but in any case, to determine whether $X <_i Y$ holds in $\hat{t}$ it is enough to know whether X contains the root of $\hat{t}$, whether Y contains the root of $\hat{t}$, and for each $j \in \{1, \ldots, r\}$ whether X is nonempty in $\hat{t}_j$, whether Y is nonempty in $\hat{t}_j$, and whether $X <_i Y$ holds in $\hat{t}_j$. These facts are determined by the set $\mathcal{R}$ and by the φ-phenotypes of $\hat{t}_j$. For the last two parts of the φ-phenotype the situation is even more direct, as X is empty in $\hat{t}$ iff it does not contain the root of $\hat{t}$ and is empty in all $\hat{t}_j$. The function $\mathrm{Comp}_{a,\varphi}$ is defined appropriately.

When $\varphi = (\psi_1 \wedge \psi_2)$ as $\mathrm{Comp}_{a,\varphi}(\mathcal{R}, \tau_1, \ldots, \tau_r)$ we take the pair of $\mathrm{Comp}_{a,\psi_i}(\mathcal{R}, \tau_1, \ldots, \tau_r)$ for $i \in \{1, 2\}$, and when $\varphi = (\neg\psi)$, we simply take $\mathrm{Comp}_{a,\varphi} = \mathrm{Comp}_{a,\psi}$.

Finally, suppose that $\varphi = \exists_{\mathsf{fin}} X.\psi$. In this situation we see that the proper definition of $\mathrm{Comp}_{a,\varphi}(\mathcal{R}, \tau_1, \ldots, \tau_r)$ is

$$\{\mathrm{Comp}_{a,\psi}(\mathcal{R} \setminus \{X\}, \sigma_1, \ldots, \sigma_r), \mathrm{Comp}_{a,\psi}(\mathcal{R} \cup \{X\}, \sigma_1, \ldots, \sigma_r) \mid \sigma_1 \in \tau_1, \ldots, \sigma_r \in \tau_r\}.$$

4 Values of Terms

In this section, we introduce the models announced in Theorem 1.1. At the end, we need the models only for sentences, but, in order to perform induction, we define them also for formulae with free variables.

Fix a formula φ of Weak MSO. We will construct a finitary model $\mathcal{M}_\varphi$ of infinitary λ-calculus, i.e., define a finite set $D_{\mathcal{M}_\varphi}[\alpha]$ for each type α, and its element $[\![K]\!]^v_{\mathcal{M}_\varphi}$ for each closed term K of type α and each $D_{\mathcal{M}_\varphi}$-valuation v with $\mathrm{dom}(v) \supseteq FV(K)$. Instead of $D_{\mathcal{M}_\varphi}[\alpha]$ and $[\![K]\!]^v_{\mathcal{M}_\varphi}$ we simply write $D_\varphi[\alpha]$ and $[\![K]\!]^v_\varphi$. When K is closed, we omit the superscript v, and we call $[\![K]\!]_\varphi$ the φ-*value* of K. Additionally, we will define a function $\mathrm{pht}_\varphi : D_\varphi[o] \to \mathrm{Pht}_\varphi$, with the intention that $\mathrm{pht}_\varphi([\![K^o]\!]_\varphi) = [BT(K)]_\varphi$.

The model is defined by induction on the formula φ. When $x = (x_1, \ldots, x_k)$ is a tuple, we write $\pi_i(x)$ for x_i.

If φ is an atomic formula, then $D_\varphi[\alpha] = \{\top\}$ and $[\![K]\!]^v_\varphi = \top$. The function pht_φ maps $\top \in \mathbb{D}_\varphi[o]$ to the unique φ-phenotype that is of the form $[t]_\varphi$ (when all set variables are mapped to the empty set, the φ-phenotype does not depend on the tree t).

If $\varphi = (\psi_1 \wedge \psi_2)$, then we take $D_\varphi[\alpha] = D_{\psi_1}[\alpha] \times D_{\psi_2}[\alpha]$, and $[\![K]\!]^v_\varphi = ([\![K]\!]^{\pi_1 \circ v}_{\psi_1}, [\![K]\!]^{\pi_2 \circ v}_{\psi_2})$, and $\mathrm{pht}_\varphi((\tau_1, \tau_2)) = (\mathrm{pht}_{\psi_1}(\tau_1), \mathrm{pht}_{\psi_2}(\tau_2))$. If $\varphi = (\neg\psi)$, then simply $D_\varphi[\alpha] = D_\psi[\alpha]$, and $[\![K]\!]^v_\varphi = [\![K]\!]^v_\psi$, and $\mathrm{pht}_\varphi = \mathrm{pht}_\psi$.

Suppose now that $\varphi = \exists_{\mathsf{fin}} X.\psi$; this case is slightly more complicated. We start by defining the domain $D_\varphi[\alpha]$, by induction on the type α, as follows:

$$D_\varphi[\alpha] = D^1_\varphi[\alpha] \times D_\psi[\alpha]\,, \qquad \text{where}$$
$$D^1_\varphi[\alpha_1 \to \cdots \to \alpha_n \to o] = (\mathrm{Pht}_\varphi)^{D_\varphi[\alpha_1] \times \cdots \times D_\varphi[\alpha_n]}\,.$$

For $\sigma \in D_\varphi[\alpha \to \beta]$ and $\tau \in D_\varphi[\alpha]$, the composition $(\sigma\,\tau) \in D_\varphi[\beta]$ is defined by

$$\pi_1(\sigma\,\tau)(\rho_1, \ldots, \rho_n) = \pi_1(\sigma)(\tau, \rho_1, \ldots, \rho_n) \qquad \text{for all arguments } \rho_1, \ldots, \rho_n, \text{ and}$$
$$\pi_2(\sigma\,\tau) = (\pi_2(\sigma)\,\pi_2(\tau))\,.$$

Next, for each term K and each φ-valuation v with $\mathrm{dom}(v) \supseteq FV(K)$, the value $[\![K]\!]^v_\varphi$ is defined using least fixpoints, as follows. A φ-*evaluation* is a function that maps every term K^α and every φ-valuation v with $\mathrm{dom}(v) \supseteq FV(K)$ to an element of $D^1_\varphi[\alpha]$. Recall that Pht_φ is a set, so the inclusion order on this set induces an order on φ-evaluations that is a complete lattice. Namely, for $\hat\sigma, \hat\tau \in D^1_\varphi[\alpha_1 \to \cdots \to \alpha_n \to o]$ we say that $\hat\sigma \subseteq \hat\tau$ if for all $\rho_1 \in D_\varphi[\alpha_1], \ldots, \rho_n \in D_\varphi[\alpha_n]$ it holds that $\hat\sigma(\rho_1, \ldots, \rho_n) \subseteq \hat\tau(\rho_1, \ldots, \rho_n)$, and for two φ-evaluations p, q we say that $p \subseteq q$ if for every term K and every φ-valuation v with $\mathrm{dom}(v) \supseteq FV(K)$ it holds that $p(K, v) \subseteq q(K, v)$. We define an operation $\mathcal{F}_\varphi$ on φ-evaluations. Let K, v and $\rho_1, \ldots, \rho_n$ be as above. We take

$$\mathcal{F}_\varphi(p)(K, v)(\rho_1, \ldots, \rho_n) = \{\mathrm{pht}_\psi([\![K]\!]^{\pi_2 \circ v}_\psi\, \pi_2(\rho_1)\, \ldots\, \pi_2(\rho_n))\} \cup \eta\,, \qquad \text{where}$$
$$\eta = \begin{cases} p(M, v)((p(N, v), [\![N]\!]^{\pi_2 \circ v}_\psi), \rho_1, \ldots, \rho_n) & \text{if } K = M\,N, \\ p(M, v[x \mapsto \rho_1])(\rho_2, \ldots, \rho_n) & \text{if } K = \lambda x.M, \\ \pi_1(v(x))(\rho_1, \ldots, \rho_n) & \text{if } K = x, \\ \mathrm{Comp}_{a,\varphi}(\emptyset, \pi_1(\rho_1), \ldots, \pi_1(\rho_n)) & \text{if } K = a. \end{cases}$$

It is clear that the operation $\mathcal{F}_\varphi$ is monotone, and hence we can consider its least fixpoint. We take $[\![K]\!]^v_\varphi = (LFP(\mathcal{F}_\varphi)(K, v), [\![K]\!]^{\pi_2 \circ v}_\psi)$, and $\mathrm{pht}_\varphi = \pi_1$.

The first component in the definition of $\mathcal{F}_\varphi(p)$, i.e. $\mathrm{pht}_\psi([\![K]\!]^{\pi_2 \circ v}_\psi\, \pi_2(\rho_1)\, \ldots\, \pi_2(\rho_n))$, intuitively corresponds to the possibility that the set X, over which we quantify, is empty

(and then we can simply read the phenotype for the subformula ψ). In the second component, η, we look inside the term, and, basically, we compose the results from its subterms. In particular, when K is a symbol ($K = a$), the call to $\mathrm{Comp}_{a,\varphi}$ takes into account both the possibility that X contains the a-labeled root, and that it does not contain it. Since we take the least fixpoint, we descend to subterms only finitely many times, and after that we have to take the first component (or end with $K = x$ or $K = a$); this corresponds to the fact that the quantification ranges over finite sets only.

We need to observe that our construction indeed satisfies conditions M1–M4 of a model. Condition M1 is clear. The other conditions are established in the following lemmata.

▶ **Lemma 4.1.** *For every term of the form $M\,N$, every formula φ of* Weak MSO*, and every φ-valuation v with $dom(v) \supseteq FV(M\,N)$ it holds that $[\![M\,N]\!]^v_\varphi = ([\![M]\!]^v_\varphi\,[\![N]\!]^v_\varphi)$.*

Proof. The proof is by induction on φ. The lemma is obvious for atomic φ, and for conjunctions and negations it follows immediately from the induction assumption. Suppose thus that $\varphi = \exists_{\mathsf{fin}} X.\psi$. The second coordinates of $[\![M\,N]\!]^v_\varphi$ and $[\![M]\!]^v_\varphi\,[\![N]\!]^v_\varphi$ contain ψ-values, equal by the induction assumption. It remains to prove for all arguments $\rho_1, \ldots, \rho_n$ that

$$\pi_1([\![M\,N]\!]^v_\varphi)(\rho_1, \ldots, \rho_n) = \pi_1([\![M]\!]^v_\varphi)([\![N]\!]^v_\varphi, \rho_1, \ldots, \rho_n)\,. \tag{1}$$

Because $\pi_1([\![\cdot]\!]^{\cdot}_\varphi)$ is a fixpoint of the $\mathcal{F}_\varphi$ operation, by the definition of $\mathcal{F}_\varphi$ we have

$$\pi_1([\![M\,N]\!]^v_\varphi)(\rho_1, \ldots, \rho_n) = \{\xi\} \cup \pi_1([\![M]\!]^v_\varphi)([\![N]\!]^v_\varphi, \rho_1, \ldots, \rho_n)\,, \qquad \text{where}$$
$$\xi = \mathrm{pht}_\psi([\![M\,N]\!]^{\pi_2 \circ v}_\psi\, \pi_2(\rho_1)\, \ldots\, \pi_2(\rho_n))\,.$$

By the induction assumption we have $[\![M\,N]\!]^{\pi_2\circ v}_\psi = [\![M]\!]^{\pi_2\circ v}_\psi\,[\![N]\!]^{\pi_2\circ v}_\psi = [\![M]\!]^{\pi_2\circ v}_\psi\,\pi_2([\![N]\!]^v_\varphi)$, Once again looking at the definition of $\mathcal{F}_\varphi$ (this time for the term M) we conclude that ξ belongs to the set $\pi_1([\![M]\!]^v_\varphi)([\![N]\!]^v_\varphi, \rho_1, \ldots, \rho_n)$, which yields (1). ◀

▶ **Lemma 4.2.** *For every term of the form $\lambda x^\alpha.M$, every formula φ of* Weak MSO*, every φ-valuation v with $dom(v) \supseteq FV(\lambda x.M)$, and every $\tau \in D_\varphi[\alpha]$ it holds that $([\![\lambda x.M]\!]^v_\varphi\,\tau) = [\![M]\!]^{v[x\mapsto\tau]}_\varphi$.*

Proof. Very similar to the previous one. ◀

▶ **Lemma 4.3.** *For every term of the form $K[M/x]$, every formula φ of* Weak MSO*, and every φ-valuation v with $dom(v) \supseteq FV(K[M/x]) \cup FV(M)$ it holds that $[\![K[M/x]]\!]^v_\varphi = [\![K]\!]^{v[x\mapsto[\![M]\!]^v_\varphi]}_\varphi$.*

Proof. The proof bases on the fact that $[\![x[M/x]]\!]^v_\varphi = [\![M]\!]^v_\varphi = [\![x]\!]^{v[x\mapsto[\![M]\!]^v_\varphi]}_\varphi$. Since $K[M/x]$ is obtained by replacing in K every x by M, and the φ-phenotypes are defined in a compositional way, using standard techniques it is easy to conclude that $[\![K[M/x]]\!]^v_\psi = [\![K]\!]^{v[x\mapsto[\![M]\!]^v_\varphi]}_\varphi$. ◀

We now come to the crucial lemma saying that the model actually computes φ-phenotypes of generated trees.

▶ **Lemma 4.4.** *For every closed term P of type o, and every formula φ of* Weak MSO *it holds that $pht_\varphi([\![P]\!]_\varphi) = [BT(P)]_\varphi$.*

Proof. The proof is by induction on φ. The lemma is obvious when φ is atomic, and it immediately follows from the induction assumption when φ is a conjunction or a negation. In the sequel we assume that $\varphi = \exists_{\mathsf{fin}} X.\psi$. In this case a φ-phenotype is a set, and thus we have to prove two inclusions. Recall that $\mathrm{pht}_\varphi = \pi_1$.

We first concentrate on the inclusion $\pi_1([\![P]\!]_\varphi) \subseteq [BT(P)]_\varphi$ (soundness). In fact, we prove a generalized statement, talking about terms of all types, and not necessarily closed; to state it, we need to say what it means for a φ-value to be sound for a term. We define it by induction on the type. Let K be a closed term of type $\alpha = \alpha_1 \to \cdots \to \alpha_n \to o$; assume that we already know which φ-values are sound for closed terms of types $\alpha_1, \ldots, \alpha_n$. We say that $\hat{\sigma} \in D^1_\varphi[\alpha]$ is *sound* for K, if whenever some $\rho_1 \in D_\varphi[\alpha_1], \ldots, \rho_n \in D_\varphi[\alpha_n]$ are sound for some closed terms $Q_1, \ldots, Q_n$, respectively, then $\hat{\sigma}(\rho_1, \ldots, \rho_n) \subseteq [BT(K\,Q_1\,\ldots\,Q_n)]_\varphi$. We say that $\sigma \in D_\varphi[\alpha]$ is *sound* for K if $\pi_1(\sigma)$ is sound for K, and $\pi_2(\sigma) = [\![K]\!]_\psi$.

Next, we define soundness for terms with free variables. We say that a φ-valuation v is *sound* for a substitution (i.e. a partial mapping from variables to closed terms) θ if $v(x)$ is sound for $\theta(x)$ for every $x \in \mathrm{dom}(v) = \mathrm{dom}(\theta)$. We say that $\sigma \in D_\varphi[\alpha]$ is *v-sound* for a term K^α, where $FV(K) \subseteq \mathrm{dom}(v)$, if whenever v is sound for a substitution θ then σ is sound for $K\theta$. When p is a φ-evaluation, we say that it is *sound* if $p(K, v)$ is v-sound for K, for all arguments K, v.

Recall that $\pi_1([\![\cdot]\!]^\cdot_\varphi)$ is the least fixpoint of the $\mathcal{F}_\varphi$ operation. We want to prove that it is sound. In order to prove some property (e.g. soundness) for the least fixpoint, it is enough to prove three things: that the property holds for the minimal element of the lattice, that it is preserved by the operation, and that it is preserved by taking the least upper bound. In our case the minimal φ-evaluation p_0 maps all arguments to the empty set, so it is clearly sound, since the empty set is a subset of every set. Consider now some set $\mathcal{P}$ of sound φ-evaluations. Its least upper bound $\bigcup \mathcal{P}$ is defined by $(\bigcup \mathcal{P})(K, v)(\rho_1, \ldots, \rho_n) = \bigcup_{p \in \mathcal{P}} p(K, v)(\rho_1, \ldots, \rho_n)$ for all arguments $K, v, \rho_1, \ldots, \rho_n$. It is thus clear that $\bigcup \mathcal{P}$ is sound, since whenever $\xi \in (\bigcup \mathcal{P})(K, v)(\rho_1, \ldots, \rho_n)$, then $\xi \in p(K)(\rho_1, \ldots, \rho_n)$ for some $p \in \mathcal{P}$. In order to obtain that the least fixpoint $\pi_1([\![\cdot]\!]^\cdot_\varphi)$ is sound, it remains to prove the following claim.

▶ Claim. If p is a sound φ-evaluation, then $\mathcal{F}_\varphi(p)$ is sound as well.

Proof. The proof is by a straightforward analysis of definitions of $\mathcal{F}_\varphi$ and of soundness. The details follow.

Let K be a term of type $\alpha = \alpha_1 \to \cdots \to \alpha_n \to o$, let v be a φ-valuation that is sound for a substitution θ, where $FV(K) \subseteq \mathrm{dom}(v)$, let $\rho_1 \in D_\varphi[\alpha_1], \ldots, \rho_n \in D_\varphi[\alpha_n]$ be sound for some closed terms $Q_1, \ldots, Q_n$ (respectively), and let $\xi \in \mathcal{F}_\varphi(p, v)(\rho_1, \ldots, \rho_n)$. Denote $t = BT(K\theta\,Q_1\,\ldots\,Q_n)$. We need to prove that $\xi \in [t]_\varphi$.

We follow the definition of $\mathcal{F}_\varphi(p)$. It is possible that $\xi = \mathrm{pht}_\psi([\![K]\!]_\psi^{\pi_2 \circ v}\,\pi_2(\rho_1)\,\ldots\,\pi_2(\rho_n))$. Then, by the definition of soundness we have that $\pi_2(\rho_i) = [\![Q_i]\!]_\psi$ for all $i \in \{1, \ldots, n\}$, and that $\pi_2(v(x)) = [\![\theta(x)]\!]_\psi$ for all $x \in FV(K)$; thus by Lemmata 4.3 and 4.1 we have $\xi = \mathrm{pht}_\psi([\![K\theta\,Q_1\,\ldots\,Q_n]\!]_\psi)$. The induction assumption of the whole lemma implies that $\xi = [t]_\psi$, and thus $\xi \in [t]_\varphi$ by the definition of a φ-phenotype of a tree.

Next, suppose that $K = M\,N$ and $\xi \in \eta = p(M, v)((p(N, v), [\![N]\!]_\psi^{\pi_2 \circ v}), \rho_1, \ldots, \rho_n)$. By assumption $p(M, v)$ is v-sound for M, hence sound for $M\theta$, and $p(N, v)$ is v-sound for N, hence sound for $N\theta$. Because $\pi_2(v(x)) = [\![\theta(x)]\!]_\psi$ for all $x \in FV(K)$, by Lemma 4.3 we have $[\![N]\!]_\psi^{\pi_2 \circ v} = [\![N\theta]\!]_\psi$, so $(p(N, v), [\![N]\!]_\psi^{\pi_2 \circ v})$ is sound for $N\theta$. Noticing that $M\theta\,(N\theta) = K\theta$, and recalling that ρ_i is is sound for Q_i, for every i, it follows from the definition of soundness that $\eta \subseteq [t]_\varphi$.

Another possibility is that $K = \lambda x.M$ and $\xi \in \eta = p(M, v[x \mapsto \rho_1])(\rho_2, \ldots, \rho_n)$. Notice that $v' = v[x \mapsto \rho_1]$ is sound for $\theta' = \theta[x \mapsto Q_1]$. By assumption $p(M, v')$ is v'-sound for M, so sound for $M\theta'$. This implies that η is a subset of the φ-phenotype of the tree generated by $M\theta'\,Q_2\,\ldots\,Q_n$, hence of t (since $M\theta'$ is β-equivalent to $K\,Q_1$).

Yet another possibility is that $K = x$ and $\xi \in \eta = \pi_1(v(x))(\rho_1, \ldots, \rho_n)$. Then, by soundness of ρ_i and $v(x)$, we obtain that η is a subset of the φ-phenotype of $BT(\theta(x)\,Q_1\,\ldots\,Q_n) = t$.

Finally, it is possible that $K = a$ and $\xi \in \eta = \mathrm{Comp}_{a,\varphi}(\emptyset, \pi_1(\rho_1), \ldots, \pi_1(\rho_n))$. Notice that the types $\alpha_1, \ldots, \alpha_n$ are o, and that n is the rank of a. By soundness of ρ_i we have that $\pi_1(\rho_i) \subseteq [BT(Q_i)]_\varphi$ for $i \in \{1, \ldots, n\}$. The root of t has label a, and the subtrees starting in its children are $BT(Q_i)$, so by monotonicity of $\mathrm{Comp}_{a,\varphi}$ we obtain

$$\xi \in \mathrm{Comp}_{a,\varphi}(\emptyset, \pi_1(\rho_1), \ldots, \pi_1(\rho_n)) \subseteq \mathrm{Comp}_{a,\varphi}(\emptyset, [BT(Q_1)]_\varphi, \ldots, [BT(Q_n)]_\varphi) = [t]_\varphi\,. \quad \blacktriangleleft$$

This finishes the proof of the „soundness" inclusion $\pi_1([\![P]\!]_\varphi) \subseteq [BT(P)]_\varphi$. We now turn into the „completeness" inclusion, i.e. $[BT(P)]_\varphi \subseteq \pi_1([\![P]\!]_\varphi)$.

Let X be a finite set of nodes of the tree generated by a term K. The following definition is by induction on the depth (distance from the root) of the deepest node in X. We say that K is *expanded up to* X if either

- X is empty, or
- $K = a\,K_1\,\ldots\,K_r$, and K_i is expanded up to the set $X{\restriction_{BT(K_i)}}$, for all $i \in \{1, \ldots, r\}$.

In other words, it is required that all nodes from X and their ancestors can be generated from K without performing any β-reductions. We prove the following claim.

▶ Claim. Let K be a closed term of type o that is expanded up to a finite set X, and let v be the valuation that maps the variable X to the set X, and all free variables of φ to the empty set. Then $[BT(K) \otimes v]_\psi \in \pi_1([\![K]\!]_\varphi)$.

Proof. The proof is by induction on the depth (distance from the root) of the deepest node in X. We have two cases. Suppose first that X is empty. Then $[BT(K) \otimes v]_\psi = [BT(K)]_\psi = \mathrm{pht}_\psi([\![K]\!]_\psi)$ by the induction assumption of the whole lemma. By definition of the $\mathcal{F}_\varphi$ operation, we see that $\mathrm{pht}_\psi([\![K]\!]_\psi)$ is an element of $\pi_1([\![K]\!]_\varphi)$, as required.

Another possibility is that $K = a\,K_1\,\ldots\,K_r$, and K_i is expanded up to the set $X{\restriction_{BT(K_i)}}$, for all $i \in \{1, \ldots, r\}$. Let v_i be the valuation that maps variable X to the set $X{\restriction_{BT(K_i)}}$, and all free variables of φ to the empty set. Lemma 3.1 says that $[BT(K) \otimes v]_\psi = \mathrm{Comp}_{a,\psi}(\mathcal{R}, [BT(K_1) \otimes v_1]_\psi, \ldots, [BT(K_r) \otimes v_r]_\psi)$, where $\mathcal{R} = \{X\}$ if X contains the root of $BT(K)$, and $\mathcal{R} = \emptyset$ otherwise. Denote $\hat{\rho}_i = \pi_1([\![K_i]\!]_\varphi)$, for all i. The induction assumption implies that $[BT(K_i) \otimes v_i]_\psi \in \hat{\rho}_i$, so by definition of $\mathrm{Comp}_{a,\varphi}$ we have $[BT(K) \otimes v]_\psi \in \mathrm{Comp}_{a,\varphi}(\emptyset, \hat{\rho}_1, \ldots, \hat{\rho}_r)$. Finally, by the definition of $\mathcal{F}_\varphi$ and by Lemma 4.1 we obtain

$$[BT(K) \otimes v]_\psi \in \mathrm{Comp}_{a,\varphi}(\emptyset, \hat{\rho}_1, \ldots, \hat{\rho}_r) \subseteq \pi_1([\![a]\!]_\varphi)([\![K_1]\!]_\varphi, \ldots, [\![K_r]\!]_\varphi) = \pi_1([\![K]\!]_\varphi)\,. \quad \blacktriangleleft$$

We come back to the proof of the inclusion $[BT(P)]_\varphi \subseteq \pi_1([\![P]\!]_\varphi)$. Take some $\xi \in [BT(P)]_\varphi$. By the definition of the φ-phenotype, $\xi = [BT(P) \otimes v]_\psi$ for some valuation v that maps the variable X to some finite set X of Σ-labeled nodes of $BT(P)$, and all free variables of φ to the empty set. It should be clear that after performing appropriately many head β-reductions from P one obtains a term P' that is expanded up to the set X (thanks to the fact that X does not contain $\bot$-labeled nodes). By the above claim we have that $\xi \in \pi_1([\![P']\!]_\varphi)$, and Fact 2.1 implies that $[\![P']\!]_\varphi = [\![P]\!]_\varphi$. This finishes the proof of the inclusion $[BT(P)]_\varphi \subseteq \pi_1([\![P]\!]_\varphi)$. ◀

4.1 Effectiveness

Thanks to Lemma 4.4 we obtain, for every sentence φ of Weak MSO, that our model recognizes the set of trees in which φ holds. In order to obtain Theorem 1.1, we need to observe that this model is effective, i.e., that the φ-value of a λY-term K can be computed from K and from φ.

This boils down to the question how to compute $[\![K]\!]_\varphi$ in the situation when $\varphi = \exists_{\mathsf{fin}} X.\psi$. Then the definition uses the least fixpoint of the $\mathcal{F}_\varphi$ operation on φ-evaluations. Although a φ-evaluation is an infinite object, we do not need to compute it for all arguments. Namely, a pair of arguments (M, v) is called *interesting*, if M is a subterm of K, and $\mathrm{dom}(v)$ contains only variables that appear in K. When (M, v) is interesting, we notice that in order to compute $\mathcal{F}_\varphi(p)(M, v)$ using the definition of $\mathcal{F}_\varphi$, it is enough to know what p returns for interesting pairs. Moreover, there are only finitely many interesting pairs: the λY-term K has only finitely many different subterms (even while being seen as an infinitary λ-term). It follows that there are only finitely many ways how a φ-evaluation may behave on interesting arguments, and thus the least fixpoint of $\mathcal{F}_\varphi$ on these arguments can be computed by repeatedly applying $\mathcal{F}_\varphi$ to the minimal φ-evaluation.

5 Conclusions

We have shown how, given a sentence of Weak MSO, one can construct an effective and finitary model of simply typed λ-calculus, that recognizes the set of trees in which the sentence is true. Whereas this result was obtained earlier in [20] by employing parity automata corresponding to Weak MSO, we define the model directly from the formula. We do not claim that our method is better or simpler, it is just different. We believe that the logical approach may be potentially useful when considering other weak logics, for which there is no natural automata model. In fact, it is an ongoing work to prove that the same result holds for the WMSO+U logic, which extends Weak MSO by a quantifier U, expressing that a subformula holds for arbitrarily large finite sets [3].

Acknowledgments. We are very grateful to Sylvain Salvati for his patience and illuminating discussions, and to reviewers of the previous version of this paper for their useful comments.

References

1 Klaus Aehlig. A finite semantics of simply-typed lambda terms for infinite runs of automata. *Logical Methods in Computer Science*, 3(3), 2007. doi:10.2168/LMCS-3(3:1)2007.

2 H. P. Barendregt. *The Lambda Calculus Its Syntax and Semantics*, volume 103. North Holland, revised edition, 1984.

3 Mikolaj Bojańczyk and Szymon Toruńczyk. Weak MSO+U over infinite trees. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th – March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 648–660. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012. doi:10.4230/LIPIcs.STACS.2012.648.

4 Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Recursion schemes and logical reflection. In *Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11-14 July 2010, Edinburgh, United Kingdom*, pages 120–129. IEEE Computer Society, 2010. doi:10.1109/LICS.2010.40.

5 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. Ordered tree-pushdown systems. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 163–177. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. URL: http://www.dagstuhl.de/dagpub/978-3-939897-97-2, doi:10.4230/LIPIcs.FSTTCS.2015.163.

6 Werner Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20(2):95–207, 1982. doi:10.1016/0304-3975(82)90009-3.

7 Charles Grellois and Paul-André Melliès. Finitary semantics of linear logic and higher-order model-checking. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*, volume 9234 of *Lecture Notes in Computer Science*, pages 256–268. Springer, 2015. doi:10.1007/978-3-662-48057-1_20.

8 Axel Haddad. Model checking and functional program transformations. In Seth and Vishnoi [22], pages 115–126. doi:10.4230/LIPIcs.FSTTCS.2013.115.

9 Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. Collapsible pushdown automata and recursion schemes. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 452–461. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.34.

10 Thomas A. Henzinger and Dale Miller, editors. *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*. ACM, 2014.

11 Martin Hofmann and Wei Chen. Abstract interpretation from Büchi automata. In Henzinger and Miller [10], pages 51:1–51:10. doi:10.1145/2603088.2603127.

12 Teodor Knapik, Damian Niwiński, and Paweł Urzyczyn. Higher-order pushdown trees are easy. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002. doi:10.1007/3-540-45931-6_15.

13 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 179–188. IEEE Computer Society, 2009. doi:10.1109/LICS.2009.29.

14 Giulio Manzonetto. Models and theories of lambda calculus. *CoRR*, abs/0904.4756, 2009. URL: http://arxiv.org/abs/0904.4756.

15 Albert R. Meyer. What is a model of the lambda calculus? *Information and Control*, 52(1):87–122, 1982. doi:10.1016/S0019-9958(82)80087-9.

16 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 81–90. IEEE Computer Society, 2006. doi:10.1109/LICS.2006.38.

17 Sylvain Salvati and Igor Walukiewicz. Evaluation is MSOL-compatible. In Seth and Vishnoi [22], pages 103–114. doi:10.4230/LIPIcs.FSTTCS.2013.103.

18 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. *Inf. Comput.*, 239:340–355, 2014. doi:10.1016/j.ic.2014.07.012.

19 Sylvain Salvati and Igor Walukiewicz. A model for behavioural properties of higher-order programs. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 229–243. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. URL: http://www.dagstuhl.de/dagpub/978-3-939897-90-3, doi:10.4230/LIPIcs.CSL.2015.229.

20 Sylvain Salvati and Igor Walukiewicz. Typing weak MSOL properties. In Andrew M. Pitts, editor, *Foundations of Software Science and Computation Structures – 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume

9034 of *Lecture Notes in Computer Science*, pages 343–357. Springer, 2015. doi:10.1007/978-3-662-46678-0_22.

21 Sylvain Salvati and Igor Walukiewicz. Using models to model-check recursive schemes. *Logical Methods in Computer Science*, 11(2), 2015. doi:10.2168/LMCS-11(2:7)2015.

22 Anil Seth and Nisheeth K. Vishnoi, editors. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2013, December 12-14, 2013, Guwahati, India*, volume 24 of *LIPIcs*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013.

23 Takeshi Tsukada and C.-H. Luke Ong. Compositional higher-order model checking via ω-regular games over Böhm trees. In Henzinger and Miller [10], pages 78:1–78:10. doi:10.1145/2603088.2603133.

On the Parallel Complexity of Bisimulation on Finite Systems

Moses Ganardi[1], Stefan Göller[2], and Markus Lohrey[1]

1 **University of Siegen, Germany**
`ganardi@eti.uni-siegen.de`
2 **Laboratoire Specification et Verification (LSV), ENS de Cachan, France; and CNRS, France**
`goeller@lsv.ens-cachan.fr`
3 **University of Siegen, Germany**
`lohrey@eti.uni-siegen.de`

Abstract

In this paper the computational complexity of the (bi)simulation problem over restricted graph classes is studied. For trees given as pointer structures or terms the (bi)simulation problem is complete for logarithmic space or NC^1, respectively. This solves an open problem from Balcázar, Gabarró, and Sántha. We also show that the simulation problem is P-complete even for graphs of bounded path-width.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases bisimulation, computational complexity, tree width

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.12

1 Introduction

Courcelle's theorem states that every problem definable in monadic second-order logic (MSO) is solvable in linear time on graphs of bounded tree-width. In recent works by Elberfeld, Jakoby, and Tantau, techniques have been developed to transfer this famous result to low space and circuit complexity classes [6, 7]. In particular, the following logspace (resp., NC^1) version of Courcelle's theorem was shown (see Section 2 for the necessary definitions):

▶ **Theorem 1** ([6, 7]). *For a fixed MSO-sentence ψ and a fixed constant k one can check in logspace whether a given structure $\mathcal{A}$ of tree-width at most k satisfies ψ. If a tree decomposition of $\mathcal{A}$ of width k is given in term representation, then one can check in* DLOGTIME*-uniform* NC^1 *whether $\mathcal{A}$ satisfies ψ.*

This result is a very powerful metatheorem, which can be applied to many computational problems. On the other hand, there are important problems solvable in logspace on graphs of bounded tree-width that are not covered by Theorem 1. One example is the graph isomorphism problem. Graph isomorphism is not MSO-definable even over finite paths since two finite paths are isomorphic if they have the same length, but one cannot express in MSO that two finite sets have the same size. Lindell [17] has shown that isomorphism of trees is in logspace, and only very recently Elberfeld and Schweitzer [8] extended this result to graphs of bounded tree-width.

In this paper, we are concerned with the complexity of simulation and bisimulation, which are of fundamental importance in the theory of reactive systems, see e.g. [1] for more

© Moses Ganardi, Stefan Göller, and Markus Lohrey;
licensed under Creative Commons License CC-BY

background. It is known that on finite state systems simulation and bisimulation are both P-complete [2], and hence have no efficient parallel algorithm unless P = NC. Surprisingly, no results on the complexity of (bi)simulation on natural subclasses of finite state systems are known (whereas there exists an extensive literature on (bi)simulation problems for various classes of infinite state systems, like pushdown systems or Petri nets). The authors of [2] pose this open question and suggest to consider the bisimulation problem on trees. The above remark that tree isomorphism cannot be expressed in MSO applies to bisimulation on trees as well (two finite paths are bisimilar if and only if they are isomorphic). Moreover, it is not clear, whether there is a natural reduction of the bisimulation problem on trees to the logspace-solvable isomorphism problem for trees (or even bounded tree-width graphs).

In this paper, we determine the complexity of the bisimulation problem on trees. More precisely, we show the following results:

- On trees the (bi)simulation problem is complete for logarithmic space (resp., NC^1) if the trees are given as pointer structures (resp., in term representation).
- The simulation problem is P-hard (and hence P-complete) already for graphs of bounded path-width.

Whether the bisimulation problem on graphs of bounded tree-width is in NC remains open.

We prove our results for the bisimulation problem by a reduction to the evaluation problem for a new class of Boolean circuits that we call *tree-shaped circuits*. These are circuits that are composed in a tree-like fashion of smaller subcircuits. We define the width of such a circuit as the maximal number of different paths from the root to an input in one of the above mentioned subcircuits. The main technical contributions of this paper are logspace- and NC^1-evaluation algorithms (depending on the input representation) for tree-shaped circuits of bounded width. These circuits should not be confused with circuits of bounded tree-width, which are known to have logspace- and NC^1-evaluation algorithms (depending on the representation) by the above Theorem 1. We show how to partially unfold tree-shaped circuits of bounded width into circuits of bounded tree-width. This unfolding is possible in TC^0 (assuming the right representation of the circuit). Finally, the resulting bounded tree-width circuit can be evaluated using Theorem 1. For the above logspace result we actually prove a stronger statement: A given tree-shaped circuit of size N and width m can be evaluated in space $O(\log N \cdot \log m)$.

One should also mention the paper [10], where a logic LREG (which extends classical first-order logic) is introduced. It is shown that LREG captures logspace on directed trees. Hence, bisimulation for trees is expressible in LREG. Due to the very technical definition of LREG we think that it is not easier to express the bisimulation problem in LREG than giving a direct logspace algorithm. Moreover expressibility in LREG does not imply that bisimulation for trees in term representation is in NC^1.

2 Preliminaries

Graphs and trees. A *(directed) graph* $G = (V, E)$ consists of a set of nodes $V(G) = V$ and a set of edges $E(G) = E \subseteq V \times V$. If $(u, v) \in E$, we call v a *successor* of u. A *path* (of length n from v_0 to v_n) in G is a node sequence $v_0, v_1, \ldots, v_n$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$. A graph is *acyclic* if there is no path of length ≥ 1 from a node to itself. We say that two nodes $u, v \in V$ are *connected* if there exists a path from u to v in the underlying undirected graph $(V, E \cup \{(v, u) \mid (u, v) \in E\})$. A set of nodes $U \subseteq V$ is *connected* if any two nodes in U are connected. The *size* $|G|$ of a graph G is the number of nodes $|V(G)|$.

A graph T is a *(rooted) tree* if there exists a node $r \in V(T)$, called the *root* of T, such that for all $v \in V(T)$ there exists exactly one path from r to v. The *depth* of a node u,

denoted by depth(u), is the length of the unique path from the root to u. A node u is an *ancestor* (resp., *proper ancestor*) of v, briefly $u \preceq v$ (resp., $u \prec v$), if there exists a path (resp., a path of length ≥ 1) in T from u to v.

A *node-labelled graph* (V, E, β) is a graph (V, E) together with a labelling function $\beta : V \to A$ into a finite set A. An *edge-labelled graph* (V, E) consists of a set of nodes V and a labelled edge relation $E \subseteq V \times A \times V$. We also write $u \xrightarrow{a} v$ instead of $(u, a, v) \in E$. Unlabelled graphs are also regarded as labelled graphs over a singleton label set. An edge-labelled tree is an edge-labelled graph (V, E) where the sets $E_a = \{(u, v) \mid (u, a, v) \in E\}$ are pairwise disjoint for $a \in A$ and $(V, \bigcup_{a \in A} E_a)$ is a directed tree.

(Bi)simulation. A *bisimulation* on an edge-labelled graph (V, E) is a binary relation $R \subseteq V \times V$ such that for all $(u, v) \in R$ the following conditions hold:

1. For all $u \xrightarrow{a} u'$ there exists $v \xrightarrow{a} v'$ such that $(u', v') \in R$.
2. For all $v \xrightarrow{a} v'$ there exists $u \xrightarrow{a} u'$ such that $(u', v') \in R$.

A relation R that only satisfies condition 1 for all $(u, v) \in R$ is called a *simulation*. A (bi)simulation on two edge-labelled graphs is a (bi)simulation on their disjoint union. Two nodes u, v are called *bisimilar* if there exists a bisimulation R such that $(u, v) \in R$. We say that u *is simulated by* v if there exists a simulation R such that $(u, v) \in R$.

It is easy to see that the union of all bisimulations is a bisimulation again and an equivalence relation, called the *bisimulation equivalence*. On finite graphs the bisimulation equivalence can be computed in polynomial time by a partition refinement algorithm [15]. In fact, the *bisimulation problem*, i.e., deciding whether two nodes are bisimilar in a given finite graph, is P-complete [2].

Tree-width and path-width. A *tree decomposition* (T, β) of a directed graph G is a node-labelled tree T, where $\beta : V(T) \to 2^{V(G)}$ assigns to each node of T a so called *bag* such that

- for all $v \in V(G)$ the set $\{t \in V(T) \mid v \in \beta(t)\}$ is non-empty and connected, and
- for all $(u, v) \in E(G)$ there exists $t \in V(T)$ such that $u, v \in \beta(t)$.

The *width* of (T, β) is $\max_{t \in V(T)} |\beta(t)| - 1$ and the *tree-width* of a graph G is the minimum width over all tree decompositions of G. A tree decomposition (T, β) is a *path decomposition* if T is a path. The *path-width* of a graph G is the minimum width of all path decompositions of G. Tree-width and path-width are also defined for node- and edge-labelled graphs via their underlying unlabelled graph. More background of tree-width can be found for instance in [5].

Circuits. A *(Boolean) circuit* $C = (G, \beta)$ is a node-labelled graph, where G is acyclic and $\beta : V(G) \to \{x_1, \ldots, x_n, 0, 1, \neg, \wedge, \vee\}$ for some n. Nodes of C are usually called *gates*. Gates labelled by 0, 1 (constant gates) or by a variable x_i (input gates) have no successors. Gates labelled by $\wedge$ or $\vee$ have at least one successor, and gates labelled by $\neg$ have exactly one successor. A *variable-free* circuit is a circuit without input gates. In a variable-free circuit every gate can be evaluated to either 0 or 1. The question, whether a given gate of a given variable-free circuit evaluates to 1 is known as the *circuit value problem*. It is one of the classical P-complete problems, see [9] for more details.

Complexity classes. In this paper we will use the complexity classes $\mathsf{AC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NC} \subseteq \mathsf{P}$. A function $f : \Sigma^* \to \Gamma^*$ is *logspace-computable* if it can be computed on a deterministic Turing-machine with a read-only input tape, a write-only output tape and a working tape whose length is bounded logarithmically in the input length. We denote by $\mathsf{L}$the

class of languages which can be decided in logspace, i.e. for which the characteristic function is logspace-computable. Throughout the paper we will use implicitly that compositions of logspace-computable functions are logspace-computable again. In the following we denote by NC^1 the class of all languages which are accepted by a DLOGTIME-uniform family of bounded fan-in Boolean circuits of polynomial size and logarithmic depth [19]. We also use the obvious generalization of this definition to functions. If we allow unbounded fan-in circuits but require constant depth, we obtain the class AC^0. If we additionally allow threshold gates, we obtain the class TC^0. It can be seen as the extension of AC^0 by the ability of counting. Typical problems in TC^0 are the computation of the sum, product and integer quotient of two binary encoded integers, and the sum and product of an arbitrary number of binary encoded integers [12]. Note that AC^0, TC^0, and NC^1 always refer to their DLOGTIME-uniform versions in this paper. For more details on space and circuit complexity we refer to [19].

Tree representations. The complexity of tree problems often depends on how the trees are represented. Firstly, trees can be given as *pointer structures* where the edge relation is given explicitly as a list of pairs consisting of two node names, which is the standard encoding of graphs in general. Secondly, trees can be given in *term representation* (or *bracket representation*): The string $()$ represents a tree of size 1. If a tree T has a root and direct subtrees $T_1, \dots, T_n$ which have term representations $r_1, \dots, r_n$, then the string $(r_1 \cdots r_n)$ is a term representation of T. Notice that a tree can have multiple term representations. Thirdly, trees can be represented in *ancestor representation* where we specify a list of all pairs (u, v) where u is an ancestor of v. Elberfeld et al. showed that term and ancestor representations can be converted into each other in TC^0 [7]. This is useful since operating on ancestor representations of trees is technically easier than on term representations.

To represent node-labelled trees (e.g., tree decompositions), in the pointer and ancestor representation we append a list of pairs consisting of a node name and a node label. In the term representation node labelled trees can be encoded by introducing for each label a an opening bracket symbol $(_a$. For instance, $(_a(_b)(_a))$ encodes a tree with an a-labelled root and two children which are labelled by b and a. If the set of node labels is not fixed (this is the case for tree decompositions) we choose an arbitrary binary block code for the opening brackets $(_a$. To represent an edge-labelled tree we transform it into a node-labelled tree, e.g. by assigning the label of an edge (u, v) to its end point v and labelling the root by a special symbol. Let us finally mention that these coding details for labelled trees are only relevant for our NC^1 lower bound in Section 4.1, which refers to AC^0-reductions. The reason is that different codings of labelled trees in term representation can be transformed in TC^0, but not necessarily in AC^0, into each other.

3 Bisimulation on Trees

In this section we consider the bisimulation problem on edge-labelled trees, i.e. the question whether the roots of two given edge-labelled trees are bisimilar. We show that the bisimulation problem is L-complete if the trees are encoded as pointer structures and NC^1-complete if the trees are given in term representation. We remark that the same complexity bounds hold for the tree isomorphism problem [4, 13, 17].

3.1 Trees as pointer structures

We start with showing a logspace upper bound for the bisimulation problem for trees in pointer representation. Bisimilarity between two edge-labelled trees T_1 and T_2 can be expressed as a

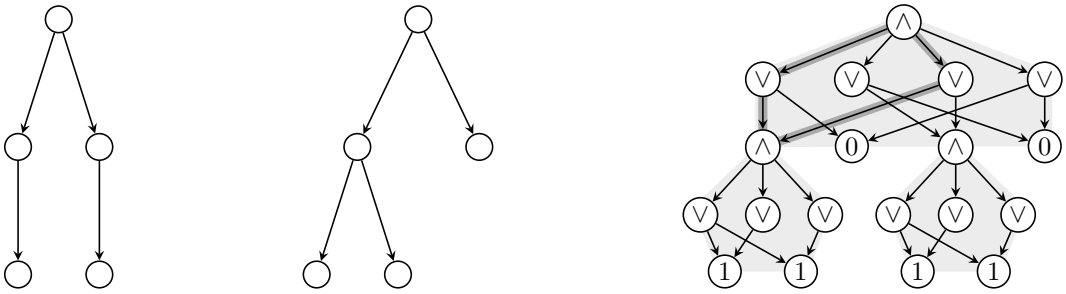


Figure 1 Two trees T_1, T_2 and the tree-shaped circuit $C(T_1, T_2)$ for bisimulation equivalence.

Boolean circuit $C(T_1, T_2)$: For all $u \in V(T_1), v \in V(T_2)$ such that $\mathrm{depth}(u) = \mathrm{depth}(v)$ the circuit contains a gate $x_{u,v}$, which evaluates to true if and only if u is bisimilar to v. We define

$$x_{u,v} = \bigwedge_{u \xrightarrow{a} u'} \bigvee_{v \xrightarrow{a} v'} x_{u',v'} \wedge \bigwedge_{v \xrightarrow{a} v'} \bigvee_{u \xrightarrow{a} u'} x_{u',v'}. \tag{1}$$

As usual we regard an empty conjunction (resp., disjunction) as 1 (resp., 0). In particular, if both u and v are leaves, then $x_{u,v} = 1$, and if exactly one of u and v is a leaf, then $x_{u,v} = 0$. An example circuit is shown in Figure 1, where T_1 and T_2 are unlabelled. Note that the circuit is composed in a tree-shaped form from smaller circuits. These smaller circuits correspond to the definition in (1) and have the crucial property that there exist exactly two paths from the root to an arbitrary leaf, which are highlighted in one subcircuit in Figure 1. This is the case because each gate variable $x_{u',v'}$ occurs once in the first conjunction and once in the second conjunction in (1). In fact, we will show that circuits with such a path property can be evaluated in logspace.

We use a more syntactic definition of such circuits: A *tree-shaped circuit* is a sequence of Boolean equations $\mathcal{S} = (x_i = \varphi_i)_{1 \le i \le n}$ where $\varphi_1, \ldots, \varphi_n$ are Boolean formulas over the variables $x_1, \ldots, x_n$ such that the graph

$$T_{\mathcal{S}} = (\{x_1, \ldots, x_n\}, \{(x_i, x_j) \mid x_j \text{ occurs in } \varphi_i\}) \tag{2}$$

is a tree with root x_1. This implies that there are no cyclic definitions in $\mathcal{S}$ and that no variable x_k appears in two different formulas φ_i and φ_j $(i \neq j)$. The *size* of $\mathcal{S}$ is defined as the sum of the sizes of all formulas φ_i. The *width* of $\mathcal{S}$ is defined as the maximal number of occurrences of a variable x_j in a formula φ_i. An example of a tree-shaped circuit of width two is given by the formulas (1) for bisimulation. We can view $\mathcal{S}$ as an ordinary Boolean circuit by taking the disjoint union of the formula trees of the φ_i and then merging all x_i-labelled leaves with the root of the formula tree of φ_i. For example, the tree-shaped circuit $C(T_1, T_2)$ for bisimulation equivalence can be regarded as a tree-shaped circuit of width 2, which can be computed in logspace from T_1 and T_2. The main goal of this section is to show that the circuit value problem restricted to tree-shaped circuits of bounded width belongs to logspace.

It is interesting to compare tree-shaped circuits of constant width with a class of circuits that is presented in [10, page 5]. For the latter, the authors require that for every path in the circuit the product of the fan-outs (i.e., indegrees, since we direct circuits towards the input gates) of the gates on the path is bounded polynomially by the circuit size. It is shown that circuits with this property can be evaluated in logspace. Note that tree-shaped circuits of constant width do not have this path property from [10]. On the other hand, the circuits

from [10] can have nodes with large fan-out, which is not possible for tree-shaped circuits of constant width. Hence, the two circuit classes are incomparable.

Note that a variable-free Boolean circuit can be represented by a relational structure with a binary edge relation and unary relations for the labels 0, 1, $\neg$, $\wedge$, $\vee$. Moreover, there is a formula of monadic-second order logic (MSO) expressing that a variable-free Boolean circuit evaluates to 1. As a consequence, by Theorem 1, we can evaluate variable-free Boolean circuits of tree-width at most k in logspace for every fixed k. Moreover, the complexity can be improved to NC^1 if we also provide for the input circuit a bounded width tree decomposition in term representation. For this one stores a tree decomposition (T, β) by an expression for the node-labelled tree T, where every node $t \in V(T)$ is labelled by the bag $\beta(t)$. Note that $\beta(t)$ is a set of gates of the circuit, and these gates are stored by their addresses. To sum up, we have:

▶ **Theorem 2.** *For every fixed $k \in \mathbb{N}$, the circuit value problem restricted to circuits of tree-width at most k can be solved in logspace. If in addition to the input circuit C a width-k tree decomposition of C in term representation is given, then the circuit value problem can be solved in* NC^1.

However, we cannot directly apply Theorem 2 to tree-shaped circuits since neither their tree-width nor clique-width is bounded by the following lemma. Clique-width is another graph measure defined by so called k-expressions, see e.g. [14] for a survey. It is easy to see that Theorem 2 can be generalized to circuits of clique-width k, provided a k-expression for the circuit is part of the input.

▶ **Lemma 3.** *For $n \geq 1$, let T_n be the tree of size $n+1$ whose root has exactly n children. The set of circuits $\{C(T_n, T_n) \mid n \geq 1\}$ has unbounded tree-width and unbounded clique-width.*

Proof. The underlying undirected graph of $C(T_n, T_n)$ contains the complete bipartite graph $K_{n,n}$ as a (topological) minor. This can be seen by removing the output gate and dissolving all constant gates (*dissolving* a node of degree 2 means deleting it and connecting its two neighbors. Figure 2 shows an example for $n = 3$. Since the tree-width of $K_{n,n}$ is n and the tree-width of every minor of a graph G is bounded by the tree-width of G, it follows that the tree-width of $C(T_n, T_n)$ is at least n.

It is known that a set of graphs which has bounded clique-width and for which there exist only finitely many n such that the bipartite graph $K_{n,n}$ is contained as a subgraph (not only minor) also has bounded tree-width [11]. We claim that for each $n \geq 1$ the undirected graph of $C(T_n, T_n)$ does not contain $K_{3,3}$ as a subgraph, which implies that $\{C(T_n, T_n) \mid n \geq 1\}$ also has unbounded clique-width (note that $C(T_n, T_n)$ contains a $K_{2,2}$, i.e., a cycle on four nodes). Note that in $C(T_n, T_n)$ for every simple path of four nodes, one of the four nodes has degree 2 (these are the gates in the middle layer of Figure 2). But this is not possible in a $K_{3,3}$. ◀

Before we show how to evaluate tree-shaped circuits of bounded width in logspace, we introduce some notions. The size $|u|$ of a node u in a tree is the size of the subtree rooted in u. We say that a node u is *heavy* if for all siblings v of u we either have (i) $|u| > |v|$ or (ii) $|u| = |v|$ and $u < v$ (where $<$ denotes some fixed order on the nodes of the tree). Otherwise a node is called *light*. Notice that the root is heavy and that every inner node has exactly one heavy child. We can compute in logspace for each node its size and determine whether it is heavy. Note that every path in a tree contains at most $O(\log n)$ light nodes.

By the following result, tree-shaped circuits of bounded width can be evaluated in logspace (take $m \in O(1)$). Also note that a tree-shaped circuit of width $m = 1$ is a tree and therefore can be evaluated in logspace.

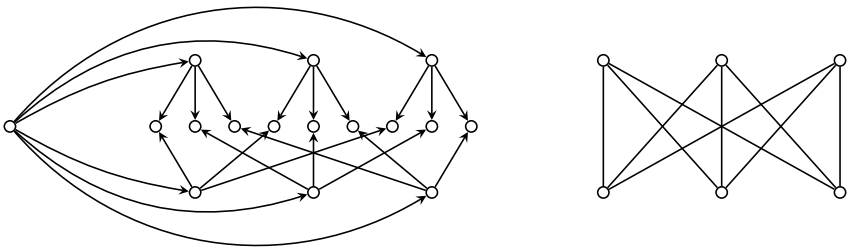


Figure 2 The circuit $C(T_3, T_3)$ for bisimulation equivalence. Removing the output gate on the left and dissolving the constant gates in the middle layer shows that $K_{3,3}$ is a minor of $C(T_3, T_3)$.

▶ **Theorem 4.** *A given tree-shaped circuit $\mathcal{S} = (x_i = \varphi_i)_{1 \le i \le n}$ of width $m \ge 2$ can be evaluated in space $O(\log s + \log n \cdot \log m)$, where $s = \max\{|\varphi_i| \mid 1 \le i \le n\}$ is the maximal size of one of the formulas φ_i. In particular, for every fixed $m \in \mathbb{N}$, there is a logspace algorithm which evaluates a given tree-shaped circuit of width at most m.*

Proof. Let $\mathcal{S} = (x_i = \varphi_i)_{1 \le i \le n}$ be a tree-shaped circuit of width m and let $s = \max\{|\varphi_i| \mid 1 \le i \le n\}$. Recall the definition of the tree $T_{\mathcal{S}}$ with node set $\{x_1, \ldots, x_n\}$ from (2).

First of all, we label every node x_i of $T_{\mathcal{S}}$ by (i) the size $|x_i|$ of the subtree rooted in x_i and (ii) a single bit indicating whether x_i is a heavy node of $T_{\mathcal{S}}$. This information can be computed in space $O(\log n)$ by traversing for each node x_i the subtree of $T_{\mathcal{S}}$ rooted in x_i in depth-first order and counting the number of nodes in binary representation.

The circuit $\mathcal{S}$ is evaluated in a recursive way using a pointer to one of the nodes $x_1, \ldots, x_n$ (which needs space $O(\log n)$) and a stack of height $O(\log n \cdot \log m)$ as follows. Initially the pointer is set to the root x_1. Assume that x_k is the heavy child of x_1 in $T_{\mathcal{S}}$. Then, the pointer is moved to x_k without writing anything on the stack. Next, the subcircuit rooted at x_k is evaluated recursively. By induction, space $O(\log s + \log n \cdot \log m)$ is used for this. Once the algorithm returns from the recursion the pointer is back on x_k and one can release the space. The value of x_k is stored on the stack, which now contains a single bit. Note that the algorithm knows (using the labeling computed before) that x_k is the heavy child of its parent node x_1. This information now triggers the evaluation of the Boolean formula φ_1. This is done by the standard evaluation algorithm that traverses the Boolean formula tree in depth-first order and stores (i) a pointer to the current node of φ_1 (this pointer needs space $O(\log |\varphi_1|)$ and (ii) a constant number of additional bits, indicating the current direction of the traversal (up or down), and the value of the current node in case we move upwards. Each time, this depth-first traversal of φ_1 arrives at a leaf node, the following is done:

- If the leaf is labelled with the variable x_k (the heavy child of x_1 in $T_{\mathcal{S}}$), then the value of x_k is retrieved from the stack and the algorithm continues the evaluation of φ_1.
- If the leaf is a light child labelled with the variable $x_i \ne x_k$, then let $1 \le c \le m$ such that the leaf corresponds to the c-th occurrence of x_i in φ_1. The algorithm stores c on the stack (which needs space $O(\log m)$) and continues recursively with the evaluation of the subcircuit rooted at x_i. Once it comes back from the recursion, the pointer is back on x_i. The algorithm sees that x_i is a light child of x_1. Using this information and the number c stored on the stack, it continues the evaluation of φ_1 at the right position in φ_1.

Note that in the second case, we have $|x_i| \le |x_1|/2$. This implies that the number of bits stored on the stack is bounded by $O(\log n \cdot \log m)$. The total space consumption is therefore $O(\log n + \log s + \log n \cdot \log m) = O(\log s + \log n \cdot \log m)$ (since we assume $m \ge 2$). ◀

Note that we do not assume $m \in O(1)$ in Theorem 4. In particular, since k, n, and m are all bounded by the size of the tree-shaped circuit (which is the sum of the sizes of the formulas φ_i), it follows that a tree-shaped circuit of size N can be evaluated in space $O(\log^2 N)$. For the special case $m \in O(1)$ (which is used for the bisimulation problem) we give an alternative proof below. This proof prepares our handling of trees in term representation in the next section (proof of Theorem 6).

Alternative proof of Theorem 4 for constant width. Let N be the size of $\mathcal{S}$ and $m \in O(1)$ its width. We will construct from $\mathcal{S}$ in logspace an equivalent polynomially sized circuit with constant tree-width, which can be seen as a partial tree unfolding of the circuit corresponding to $\mathcal{S}$. By Theorem 2 the resulting circuit can be evaluated in logspace.

As before, we view the Boolean formulas φ_i as labelled trees. For simplicity we assume that all φ_i have at least size two, which can be ensured by replacing φ_i by $\varphi_i \wedge 1$, so that $\mathcal{S}$ contains no "chain rules". Moreover, we assume that the trees φ_i have disjoint node sets. Let x_i be an inner node of $T_{\mathcal{S}}$ whose heavy child is x_k. Inductively we define a circuit C_i as follows: We take the formula φ_i, viewed as a tree, and merge all x_k-labelled leaves into a single node. Note that x_j-labelled leaves for $j \neq k$ are not merged. Then we insert into each leaf labelled by some variable x_j a copy of the circuit C_j. Finally let C be C_1, which clearly evaluates to the same truth value as $\mathcal{S}$. Figure 3 shows the circuit resulting from the circuit $C(T_1, T_2)$ on the right in Figure 1.

Note that the number of copies of C_i in C is bounded by m^{ℓ_i} where ℓ_i is the number of light nodes on the path from x_1 to x_i in $T_{\mathcal{S}}$. Since $\ell_i \leq \log n$, C has size at most $m^{\log n} \cdot N$, which is bounded by $n^{O(1)} \cdot N$ since m is a constant.

Furthermore C can be computed in logspace from $\mathcal{S}$. To make this explicit, we introduce a naming scheme for the gates in C. The set $\mathrm{Addr}(x_i)$ (addresses for the copies of x_i) contains finite words over the alphabet $\{1, \ldots, m\}$ defined inductively: We set $\mathrm{Addr}(x_1) = \{\varepsilon\}$ for the root x_1. If x_j is the heavy child of x_i in $T_{\mathcal{S}}$, we set $\mathrm{Addr}(x_j) = \mathrm{Addr}(x_i)$. If x_j is a light child of x_i in $T_{\mathcal{S}}$, we set $\mathrm{Addr}(x_j) = \mathrm{Addr}(x_i) \cdot \{1, \ldots, k\}$ where $k \leq m$ is the number of occurrences of x_j in φ_i. The sets $\mathrm{Addr}(x_i)$ contain words of length $O(\log n)$ over the constant sized alphabet $\{1, \ldots, m\}$. Moreover, given a word of length $O(\log n)$ over $\{1, \ldots, m\}$ we can easily check in logspace whether it belongs to $\mathrm{Addr}(x_i)$ by traversing the path from x_i to the root of $T_{\mathcal{S}}$. Now we can define the circuit C over the gate set

$$V(C) = \bigcup_{i=1}^{n} \{(u, a) \mid a \in \mathrm{Addr}(x_i),\ u \text{ is a non-input node of the tree } \varphi_i\}.$$

For every edge (u, u') of φ_i $(1 \leq i \leq n)$ and $a \in \mathrm{Addr}(x_i)$, we add the following edges to $E(C)$:

1. If u' is a non-input gate, add the edge $((u, a), (u', a))$.
2. If u' is labelled by x_j, let v be the root of φ_j and add the edge
   a. $((u, a), (v, a \cdot d))$ if x_j is a light child of x_i and u' is its d-th occurrence of x_j in φ_i,
   b. $((u, a), (v, a))$ if x_j is the heavy child of x_i.

The labels of the gates in C are inherited from the formula trees φ_i. Note that a pair (u, a) from $V(C)$ can be stored in logspace. Moreover, whether a pair belongs to $V(C)$ and whether a pair of nodes from $V(C)$ belongs to $E(C)$ can be checked in logspace. Hence, the circuit C can be constructed in logspace.

Finally we show that the tree-width of C is at most 2. Consider the subgraph T of C where edges of type 2b in the above definition of C are removed if u' is the d-th occurrence of x_j in φ_i for some $d > 1$. In Figure 3 such edges are drawn as dotted lines. The resulting

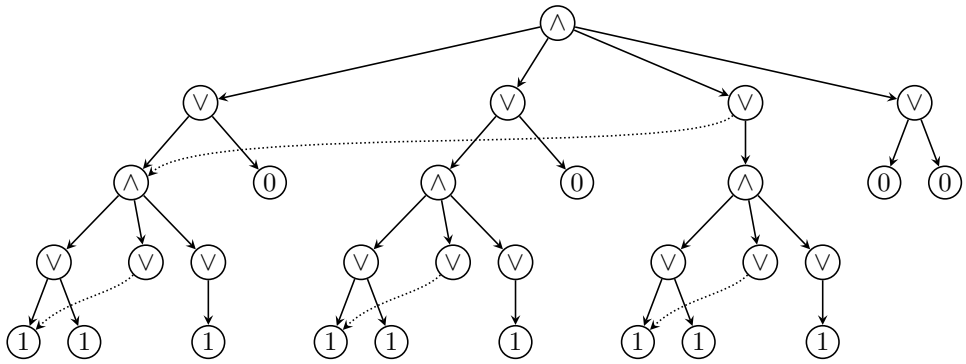


Figure 3 The partial unfolding of the tree-shaped circuit $C(T_1, T_2)$ from Figure 1.

subgraph T is indeed a tree, on which we define a tree decomposition (T, β) of C. Let u be a non-input node of φ_i and $a \in \mathrm{Addr}(x_i)$. The bag $\beta(u, a)$ contains the gate (u, a), its parent node in the tree T (if existent) and the unique node (v, a) where v is the root of φ_j and x_j is the heavy child of x_i (if existent). One can verify that (T, β) is indeed a valid tree decomposition. ◀

We can apply Theorem 4 to the tree-shaped circuit defined by (1) to solve the tree bisimulation problem in logspace:

▶ **Corollary 5.** *The bisimulation problem for trees given as pointer structures is in* L.

3.2 Trees in term representation

Next we will show that the tree bisimulation problem belongs to NC^1 if the trees are given as terms. For that we prove an NC^1-version of Theorem 4 (for $m \in O(1)$), which uses the NC^1-part of Theorem 2, where a tree decomposition in term representation is part of the input. Here we require that the tree-shaped circuit $\mathcal{S} = (x_i = \varphi_i)_{1 \le i \le n}$ must be given together with the underlying tree $T_{\mathcal{S}}$ in term representation. We also assume that the Boolean formulas φ_i are given in term representation. Recall the ancestor representation of a tree from Section 2 and that it can be transformed into the term representation of a tree in TC^0 and vice versa.

▶ **Theorem 6.** *For every fixed $m \in \mathbb{N}$, one can evaluate in NC^1 a given tree-shaped circuit $\mathcal{S} = (x_i = \varphi_i)_{1 \le i \le n}$ (with all φ_i given in term representation) of width at most m that is given together with the term representation of the tree $T_{\mathcal{S}}$.*

Before we prove Theorem 6, we apply it to the tree bisimulation problem.

▶ **Theorem 7.** *The bisimulation problem for trees given in term representations is in* NC^1.

Proof. First we convert the term representations of T_1 and T_2 into ancestor representations in TC^0. We can compute the tree-shaped circuit $\mathcal{S}$ corresponding to the circuit $C(T_1, T_2)$ and an ancestor representation of $T_{\mathcal{S}}$ in TC^0. The set of variables $\{x_{u,v} \mid u \in V(T_1), v \in V(T_2), \mathrm{depth}(u) = \mathrm{depth}(v)\}$ can be clearly computed in TC^0 since for a given node of a tree in ancestor representation one can compute its depth by counting ancestors. The term representations of the formulas $\varphi_{u,v}$ in (1) can then be computed in AC^0. The ancestor representation of $T_{\mathcal{S}}$ is also AC^0-computable, since $x_{u,v}$ is an ancestor of $x_{u',v'}$ if and only if u is an ancestor of u' and v is an ancestor of v'. By Theorem 6 we can evaluate $\mathcal{S}$ in NC^1. ◀

Proof of Theorem 6. Let $\mathcal{S} = (x_i = \varphi_i)_{1 \leq i \leq n}$ be a tree-shaped circuit of width at most m, where every φ_i is given in term representation. Moreover, we assume to have the term (or ancestor) representation of the tree $T_{\mathcal{S}}$. We will show how to compute in TC^0 the partial unfolding C of $\mathcal{S}$ and the tree decomposition of C in ancestor representation from the alternative proof of Theorem 4 on page 12:8.

Let N be the size of $\mathcal{S}$. For each node x_i of $T_{\mathcal{S}}$ we can compute in TC^0 its depth (by counting ancestors) and the size of the subtree below x_i (by counting descendants). Hence, we can also compute the heavy child of every inner node x_i in $T_{\mathcal{S}}$. Additionally, if x_i has a parent node x_j, we can compute the number $d(x_i) \in \{1, \ldots, m\}$ of occurrences of x_i in φ_j. We transform all formulas φ_i of $\mathcal{S}$ into ancestor representation in TC^0 where we can assume that the encodings of all nodes have length $O(\log N)$. Also we can ensure that all formulas have at least size two.

Recall that every node u of a formula φ_i has multiple copies (u, a) in C where $a \in \mathrm{Addr}(x_i)$ is an address of length $O(\log n)$ defined on page 12:8. Given a string $a \in \{1, \ldots, m\}^*$ of length $O(\log n)$ and a node x_i we can verify in TC^0 whether $a \in \mathrm{Addr}(x_i)$: From the ancestor representation of $T_{\mathcal{S}}$ we can compute the sequence of all light nodes $x_{i_1} \prec x_{i_2} \prec \cdots \prec x_{i_k}$ in $T_{\mathcal{S}}$ on the path from the root to x_i. This can be done by sorting all light ancestors of x_i by their depth in ascending order. It is known that sorting n numbers with n bits each is in TC^0 [19]. Then we have $a_1 \cdots a_k \in \mathrm{Addr}(x_i)$ if and only if $a_j \leq d(x_{i_j})$ for all $j \in \{1, \ldots, k\}$. Hence, we can also encode all gates of the partial unfolding C by strings of length $O(\log N)$ and can compute $V(C)$ in TC^0. With the previous preparation the edge relation $E(C)$ can be computed in AC^0 using the definition on page 12:8.

It remains to show that we can compute in TC^0 the ancestor representation of the width-2 tree decomposition (T, β) of C from page 12:9. We set $V(T) = V(C)$. Let $(u, a), (u', a') \in V(T)$ be nodes where u (resp., u') belongs to φ_i (resp., φ_j). Then (u, a) is an ancestor of (u', a') in T if and only if $x_i \preceq x_j$ in $T_{\mathcal{S}}$, a is a prefix of a' and the following holds:

1. If $x_i = x_j$, then $a = a'$ and $u \preceq u'$ in φ_i.
2. If $x_i \prec x_j$ in $T_{\mathcal{S}}$, let x_k be the unique child of x_i which is an ancestor of x_j.
   a. If x_k is a light node, let $d \in \{1, \ldots, m\}$ be the number in a' at position $|a| + 1$. Then the d-th occurrence of x_k in φ_i is a descendant of u.
   b. If x_k is a heavy node, then the first occurrence of x_k in φ_i is a descendant of u.

The last condition forbids those edges that were deleted when constructing the tree T from the partial unfolding C (the dotted edges in Figure 3). The bag-function β can be computed straightforward in TC^0 using its definition on page 12:9. By Theorem 2 we can evaluate C in NC^1, which concludes the proof. ◀

Equality of hereditarily finite sets. The bisimulation problem on trees in term representation arises in a very natural way. A *hereditarily finite set* is either the empty set $\{\}$ or a set $\{a_1, \ldots, a_n\}$ containing finitely many hereditarily finite sets $a_1, \ldots, a_n$. Hereditarily finite sets have a natural string representation over the bracket symbols { and }. By counting brackets, one can check in TC^0, whether a string over { and } is well-bracketed [3]. As before, such a well-bracketed string corresponds to a tree. By induction over the height of trees, one can easily show that two well-bracketed strings over { and } represent the same set if and only if the corresponding trees are bisimilar. Hence, the tree bisimulation problem for (unlabelled) trees in term representation is equivalent to the *set equality problem*, which asks whether two such string representations represent the same set. For example {{}{}} and {{}} represent the same set. From Theorem 7 we obtain the NC^1 upper bound in the following result. The NC^1 lower bound follows from the NC^1-hardness of the bisimulation problem for unlabelled trees in term representation, which is shown in the next section (Theorem 9).

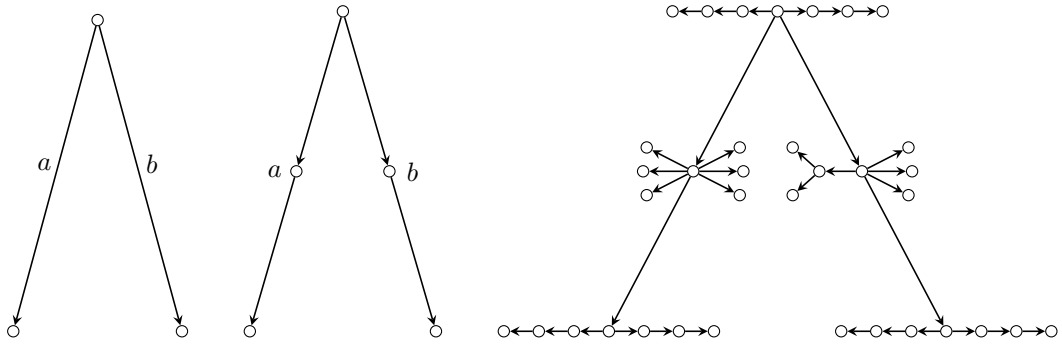


Figure 4 From labelled to unlabelled trees.

▶ **Corollary 8.** *The set equality problem is* NC^1*-complete with respect to* AC^0*-reductions.*

4 Lower Bounds

In this section we prove several lower bounds. In Section 4.1 we prove matching lower bounds for the upper bounds from Corollary 5 and Theorem 7. Finally, in Section 4.2 we show that the simulation problem becomes already P-complete for graphs of bounded path-width.

4.1 Bisimulation on Trees

▶ **Theorem 9.** *The bisimulation problem for unlabelled trees is* L*-hard if the trees are given as pointer structures and* NC^1*-hard if they are given in term representation (both with respect to* AC^0*-reductions).*

Before we prove Theorem 9 let us first show the following lemma:

▶ **Lemma 10.** *The bisimulation problem for edge-labelled trees in term representation (resp., pointer representation) is* AC^0*-reducible to the bisimulation problem for unlabelled trees in term representation (resp., pointer representation).*

Proof. We only show the lemma for the term representation; the same construction also works for the pointer representation. In [18] Srba presents a reduction from the bisimulation problem for edge-labelled graphs to the bisimulation problem for unlabelled graphs. In fact, this construction transforms trees into trees. We slightly modify the reduction to ensure AC^0-computability and assume that there are only two labels, say a and b (which is the case for the trees constructed in the proof of Theorem 9).

Consider a tree T with edge labels a and b. First every labelled edge of T is subdivided into two edges. In Figure 4 (middle tree), the new node added for an x-labelled edge ($x \in \{a, b\}$) is labelled with x. To distinguish the original nodes from the new nodes, we attach to each original node two paths of length 3. To each new node we attach one of two small trees depending on the label of the original edge that is represented by the new node, see the right tree in Figure 4. Let us denote the resulting unlabelled tree with $\mathsf{ul}(T)$. It is not hard to prove that two labelled trees T_1 and T_2 are bisimilar if and only if $\mathsf{ul}(T_1)$ and $\mathsf{ul}(T_2)$ are bisimilar. The proof is basically given in [18].

It remains to prove that the term representation of $\mathsf{ul}(T)$ can be computed in AC^0 from the term representation of T. Consider the term representation t of T. Recall that we identify the opening brackets in t with the nodes of T. We assume that the term representation t

contains the following three opening bracket types: $(_a$ and $(_b$, which represent nodes with an incoming a-labelled (resp. b-labelled) edge, and $($ for the root. For example, $t = ((_a)(_b))$ is a term representation for the left tree in Figure 4. The transformation $T \mapsto \mathsf{ul}(T)$ can be described by two isometric homomorphisms. A homomorphism $h : \Sigma^* \to \Gamma^*$ is *isometric* if there is an $\ell \geq 1$ such that $|h(c)| = \ell$ for all $c \in \Sigma$. In [16] it is shown that for a given isometric homomorphism $h : \Sigma^* \to \Gamma^*$ and a word $w \in \Sigma^*$ one can compute $h(w)$ in AC^0.[1]

We will proceed in two steps. Define the isometric homomorphism $h_1 : \{(_a, (_b,)\}^* \to \{(_a, (_b, (,),]\}^*$ by:

$$(_a \mapsto (_a(\qquad (_b \mapsto (_b(\qquad) \mapsto)]$$

Let $u \in \{(_a, (_b,)\}^*$ be the word such that $t = (u)$ and consider the string $(h_1(u))$. Formally, it is not a term representation (since we have two types of closing brackets). Nevertheless, it describes the tree obtained from T by subdividing every edge and labelling each new node with the former edge label. For example $t = ((_a)(_b))$ is transformed into $(h_1(u)) = ((_a()](_b()])$, which describes the node-labelled tree in Figure 4. The second isometric homomorphism $h_2 : \{(_a, (_b, (,),]\}^* \to \{(,)\}^*$ is defined by:

- $(\mapsto ((((()))$ (an opening bracket followed by a path of length 3)
- $) \mapsto ((()))))$ (a path of length 3 followed by a closing bracket)
- $] \mapsto ()()())$ (3 leaves followed by a closing bracket)
- $(_a \mapsto (()()()$ (an opening bracket followed by 3 leaves)
- $(_b \mapsto (((())$ (an opening bracket followed by the tree)

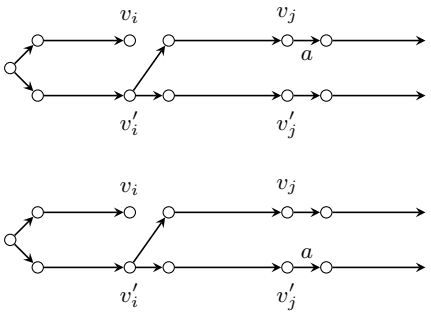
Then, the string $h_2((h_1(u)))$ is indeed a term representation for the desired unlabelled tree $\mathsf{ul}(T)$. ◀

Proof of Theorem 9. By Lemma 10 it suffices to show the lower bounds for edge-labelled trees. We reuse the proofs from [13], where it is shown that the tree isomorphism problem is L-hard (NC^1-hard, respectively) with respect to AC^0-reductions if the the trees are given as pointer structures (in term representation, respectively). Let us start with the bisimulation problem for trees given in pointer representation. Here, Jenner et al. reduce from the L-complete reachability problem on paths, i.e., the question whether for a given directed path graph G and two nodes $v_i, v_j \in V(G)$, there is a path from v_i to v_j. Without loss of generality, v_i and v_j are distinct and have successors v_{i+1} and v_{j+1}, respectively.

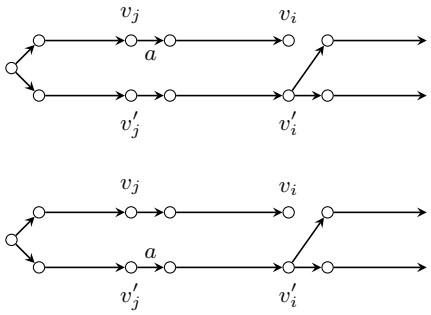
Consider the tree with a root node which has two copies of G as direct subtrees. We refer to nodes of the two copies by $v_1, \ldots, v_n$ and $v'_1, \ldots, v'_n$, respectively. Additionally we replace the edge (v_i, v_{i+1}) by the new edge (v'_i, v_{i+1}). Now let T_1 (resp., T_2) be the tree where the edge (v_j, v_{j+1}) (resp., (v'_j, v'_{j+1})) is labelled by a symbol a (all unlabelled edges are assumed to be labelled with a symbol $b \neq a$). Clearly, T_1 and T_2 can be computed in AC^0 from G. There is a path from v_i to v_j in G if and only if T_1 and T_2 are bisimilar. See Figure 5 for an illustration of the reduction.

Secondly, Jenner et al. present in [13] an AC^0-reduction from the NC^1-complete evaluation problem of balanced Boolean expressions to the isomorphism problem for trees in term representation. They use the AND-gadget $T_\wedge(G_1, G_2, H_1, H_2)$ and the OR-gadget $T_\vee(G_1, G_2, H_1, H_2)$ which are depicted in Figure 6. Notice that for all trees G_1, G_2, H_1, H_2 the following holds, where $\sim$ can both mean bisimilarity and isomorphism:

[1] If the homomorphism is fixed, this is even possible in NC^0. Moreover, if the homomorphism is not isometric then the problem is TC^0-complete [16].

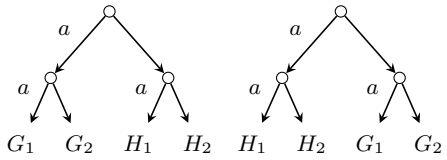


(a) T_1 and T_2 are bisimilar.

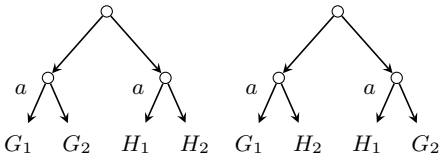


(b) T_1 and T_2 are not bisimilar.

Figure 5 The two possible forms of T_1, T_2 depending on whether v_j is reachable from v_i or not.



(a) AND-gadget $T_\wedge$



(b) OR-gadget $T_\vee$

Figure 6 The trees for the NC^1 lower bound.

- $G_1 \sim H_1$ and $G_2 \sim H_2 \iff T_\wedge(G_1, G_2, H_1, H_2) \sim T_\wedge(H_1, H_2, G_1, G_2)$, and
- $G_1 \sim H_1$ or $G_2 \sim H_2 \iff T_\vee(G_1, G_2, H_1, H_2) \sim T_\vee(G_1, H_2, H_1, G_2)$.

Using the fact that the AND-gadget and the OR-gadget have the same tree structure (if we ignore labels), one can show that the term representations for the resulting trees can be computed in AC^0 from the balanced Boolean expression; see the arguments in [13]. This yields an AC^0-reduction from the evaluation problem of balanced Boolean expressions to the bisimulation problem for trees in term representation. ◀

4.2 Simulation on Graphs of Bounded Path-Width

The results from Corollary 5 and Theorem 7 also hold for the simulation problem. The proof is in fact much easier, since the simulation problem for trees reduces to the evaluation of the Boolean circuit obtained from (1) by removing the second conjunction over all edges $v \xrightarrow{a} v'$; in fact, this circuit is a tree. In this section we show that the simulation problem is P-complete on graphs of bounded path-width, and hence also on graphs of bounded tree-width. It remains open, whether the bisimulation problem for graphs of bounded tree-width belongs to NC or remains P-complete. For integers i, j we use the abbreviation $[i, j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$.

▶ **Theorem 11.** *There is a number k such that the following problem is* P*-complete: Given a graph G of path-width at most k and two nodes $u, v \in V(G)$, does u simulate v?*

Proof. Fix a P-complete language $L \subseteq \{0,1\}^*$ and a deterministic polynomial time bounded Turing machine $M = (Q, \Gamma, \{0,1\}, q_0, q_f, \delta)$ that accepts L. Here Q is the set of states, $\Gamma \supseteq \{0, 1, \Box\}$ is the tape alphabet ($\Box$ is the blank symbol), q_0 is the initial state, q_f is the final state, and $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\rightarrow, \leftarrow\}$ is the transition function, where $\rightarrow$ and $\leftarrow$ indicate the head direction. The machine has a single tape, whose cells are indexed with

integers. Initially, the input x is written in cells $0, \ldots, |x|-1$ and the tape head scans cell 0. We can assume that there is a polynomial $p(n)$ such that for every input $x \in \{0,1\}^*$ we have: $x \in L$ if and only if after $p(|x|)$ many transitions the machine is in state q_f, cell 0 contains $\Box$, and the tape head scans cell 0.

We can view configurations of M as words from $\Gamma^*(Q \times \Gamma)\Gamma^*$. Let $\Omega = \Gamma \cup (Q \times \Gamma)$. We define a partial mapping $\Delta : \Omega^3 \to \Omega$ as follows, where $a, a', b, c \in \Gamma$, $p, q \in Q$.

- $\Delta(a,b,c) = b$
- $\Delta(b,c,(q,a)) = (p,c)$ if $\delta(q,a) = (p,a',\leftarrow)$
- $\Delta(b,c,(q,a)) = c$ if $\delta(q,a) = (p,a',\rightarrow)$
- $\Delta(b,(q,a),c) = a'$ if $\delta(q,a) = (p,a',d)$ for some $d \in \{\rightarrow,\leftarrow\}$
- $\Delta((q,a),b,c) = b$ if $\delta(q,a) = (p,a',\leftarrow)$
- $\Delta((q,a),b,c) = (p,b)$ if $\delta(q,a) = (p,a',\rightarrow)$

In all other cases, Δ is undefined. The mapping Δ computes from the three symbols at positions $i-1$, i, $i+1$ in a configuration the symbol at position i in the successor configuration.

Let us fix an input $x = a_0 a_1 \cdots a_{n-1}$ of length $n > 0$ for the machine M and let $N = p(n)+1$. Then there exists a unique computation of M on input x. We denote with C the corresponding computation table. Formally, it is a mapping $C : [-N,N] \times [0,N-1] \to \Omega$, where $C(i,t)$ is the symbol at cell i in the t-th configuration. It can be defined by the following properties:

- $C(0,0) = (a_0,q_0)$, $C(i,0) = a_i$ for $i \in [1,n-1]$, $C(i,0) = \Box$ for $i \in [-N,N] \setminus [0,n-1]$,
- $C(-N,t) = C(N,t) = \Box$ for all $t \in [0,N-1]$
- $C(i,t) = \Delta(C(i-1,t-1), C(i,t-1), C(i+1,t-1))$ for all $t \in [1,N-1]$, $i \in [-N+1,N-1]$.

Let us fix the set of edge labels $A = \{-1,0,1,\alpha\} \uplus \Omega$. We define two edge-labelled graphs P (for position) and T (for time) with edge labels from A and the node sets

$$V(P) = [-N,N] \times \{0,1\}, \quad V(T) = [0,N-1] \;\cup\; [0,N-1] \times \Omega \;\cup\; [0,N-1] \times \Omega^3.$$

For better readability, we write edges of P (resp., T) as $x \xrightarrow{a}_P y$ (resp., $x \xrightarrow{a}_T y$). Then, P and T contain the following edges:

$(i,0) \xrightarrow{\alpha}_P (i,1)$ for all $i \in [-N,N]$

$(i,1) \xrightarrow{\delta}_P (i+\delta,0)$ for all $i \in [-N,N], \delta \in \{-1,0,1\}$ with $i+\delta \in [-N,N]$

$(i,0) \xrightarrow{C(i,0)}_P (i,0)$ for all $i \in [-N,N]$

$(t,a) \xrightarrow{\alpha}_T (t-1,b,c,d)$ for all $a \in \Omega, t \in [1,N-1], (b,c,d) \in \Delta^{-1}(a)$

$(t,a_{-1},a_0,a_1) \xrightarrow{\delta}_T (t,a_\delta)$ for all $t \in [0,N-2], a_{-1},a_0,a_1 \in \Omega, \delta \in \{-1,0,1\}$

$(t,a) \xrightarrow{b}_T t$ for all $a,b \in \Omega, t \in [1,N-1]$

$(0,a) \xrightarrow{b}_T 0$ for all $a \in \Omega, b \in \{a,\alpha\}$

$t \xrightarrow{a}_T t$ for all $t \in [0,N-1], a \in A$

An example of the construction is shown in Figure 7, where we assume $N = 3$ for simplicity. It is easy to see that both P and T (and hence also the disjoint union of P and T) have bounded path-width. More precisely, P has path-width 3 (the bags are the sets $\{(i,0),(i,1),(i+1,0),(i+1,1)\}$ for $-N \le i \le N-1$), whereas the path-width of T is bounded by $|\Omega|^3 + |\Omega|$ (the bags are the set $\{t,(t,a),(t-1,b,c,d) \mid a,b,c,d \in \Omega\}$ for $1 \le t \le N-1$ and $\{(t,a),(t,b,c,d) \mid a,b,c,d \in \Omega\}$ for $0 \le t \le N-2$). Recall that $|\Omega|$ is a fixed constant since the machine M is fixed.

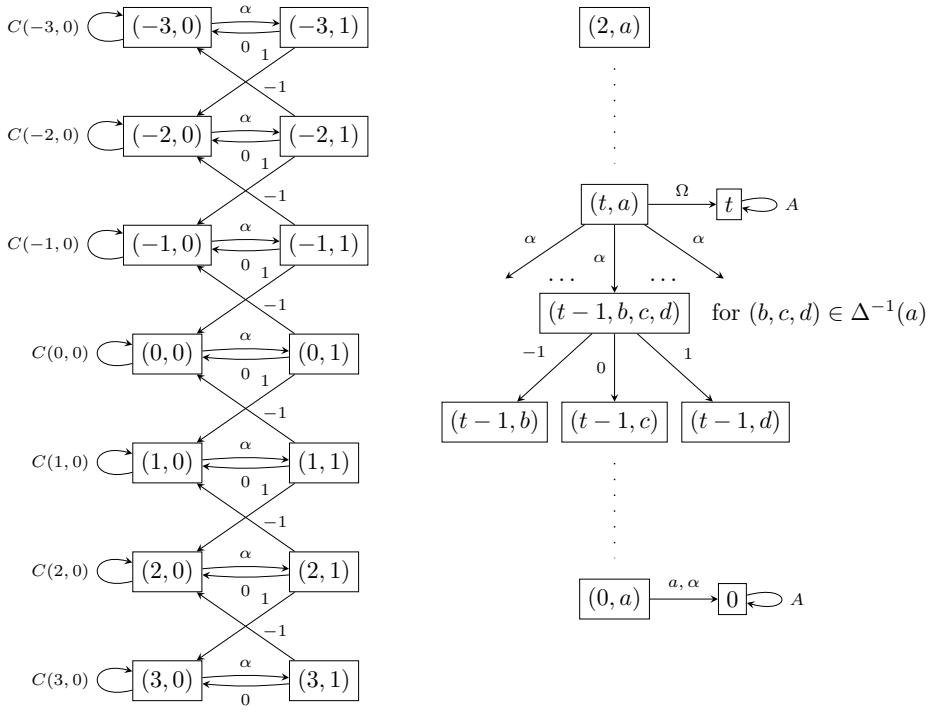


Figure 7 The edge-labelled graphs P and T, where we assume $N = 3$ for simplicity.

For nodes $u \in V(P)$, $v \in V(T)$ we write $u \sqsubseteq v$ if u is simulated by v in the disjoint union of P and T. The following claim proves the theorem, since M accepts x if and only if $C(0, N-1) = (q_f, \Box)$ by our assumptions on M.

▶ **Claim.** *For all $t \in [0, N-1]$, $i \in [-N, N]$ and $a \in \Omega$ such that $i + t \leq N$ and $i - t \geq -N$ we have: $C(i,t) = a$ if and only if $(i, 0) \sqsubseteq (t, a)$.*

Proof of Claim. We prove the claim by induction on t. First, note that for every $v \in V(P)$ and every $t \in [0, N-1] \subseteq V(T)$ we have $v \sqsubseteq t$, since at t we can loop with every $a \in A$. For $t = 0$ note that indeed $(i, 0) \sqsubseteq (0, C(i, 0))$: The only outgoing edges for $(i, 0)$ are labelled with $C(i, 0)$ and α. From $(0, C(i, 0))$ these labels lead to node 0, which simulates every node of P. On the other hand, if $a \neq C(i, 0)$, then $(i, 0)$ has a $C(i, 0)$-labelled outgoing edge, whereas $(0, a)$ has no such outgoing edge. This implies $(i, 0) \not\sqsubseteq (0, a)$.

Now assume that $t \in [1, N-1]$, $i \in [-N, N]$, $i + t \leq N$, and $i - t \geq -N$ and that the claim holds for $t - 1$. First assume that $C(i, t) = a$. Since $t \geq 1$, we have $i + 1 \leq N$ and $i - 1 \geq -N$, i.e., $i \in [-N + 1, N - 1]$. We have to show that $(i, 0) \sqsubseteq (t, a)$. The edge $(i, 0) \xrightarrow{C(i,0)}_P (i, 0)$ can be simulated by the edge $(t, a) \xrightarrow{C(i,0)}_T t$ (recall that $(i, 0) \sqsubseteq t$). Now consider the other possible edge $(i, 0) \xrightarrow{\alpha}_P (i, 1)$. Since $C(i, t) = a$, there must exist $(b, c, d) \in \Delta^{-1}(a)$ such that $b = C(i - 1, t - 1)$, $c = C(i, t - 1)$, and $d = C(i + 1, t - 1)$. Also note that $i + \delta + (t - 1) \leq N$ and $i + \delta - (t - 1) \geq -N$ for all $\delta \in \{-1, 0, 1\}$. Hence, by induction $(i - 1, 0) \sqsubseteq (t - 1, b)$, $(i, 0) \sqsubseteq (t - 1, c)$, and $(i + 1, 0) \sqsubseteq (t - 1, d)$. But this implies that $(i, 1) \sqsubseteq (t - 1, b, c, d)$. Hence, we can choose the edge $(t, a) \xrightarrow{\alpha}_T (t - 1, b, c, d)$ in order to simulate the edge $(i, 0) \xrightarrow{\alpha}_P (i, 1)$.

Finally, assume that $C(i, t) \neq a$. We have to show that $(i, 0) \not\sqsubseteq (t, a)$. Let us choose the edge $(i, 0) \xrightarrow{\alpha}_P (i, 1)$. We have to show that for every $(b, c, d) \in \Delta^{-1}(a)$, $(i, 1) \not\sqsubseteq (t, b, c, d)$.

Let us fix a triple $(b, c, d) \in \Delta^{-1}(a)$. Since $C(i, t) \neq a$, one of the following three statements holds: $C(i-1, t-1) \neq b$, $C(i, t-1) \neq c$, $C(i+1, t-1) \neq d$. Hence, by induction, $(i-1, 0) \not\sqsubseteq (t-1, b)$ or $(i, 0) \not\sqsubseteq (t-1, c)$ or $(i+1, 0) \not\sqsubseteq (t-1, d)$. This implies that, indeed, $(i, 1) \not\sqsubseteq (t, b, c, d)$. ◀
◀

It seems to be difficult to modify the above proof so that it shows P-hardness for bisimulation on graphs of bounded path-width or bounded tree-width. One might try to restrict the choices of the players in the bisimulation game (see e.g. [1]) so that they are forced to play as in the simulation game. There is a technique to achieve this (defenders forcing) but the problem is that it yields grid-like graph structures and hence graphs of unbounded tree-width.

5 Conclusion

We proved the following results:

- The bisimulation problem for trees that are given by pointer structures (resp., in term representation) is complete for deterministic logspace (resp. NC^1). These results also hold for the simulation problem for trees.
- Already for graphs of bounded path-width (a subclass of the graphs of bounded tree-width), the simulation problem becomes P-complete.

As an application of the first result, we showed that equality of hereditarily finite sets is NC^1-complete. For the proofs we introduced the new class of tree-shaped circuits and proved that the circuit evaluation problem for tree-shaped circuits of bounded width is in logspace or NC^1, depending on the representation of the circuit. It would be nice to find further applications of these circuits. The main open problem that remains is whether the bisimulation problem for graphs of bounded tree-width is in NC or P-complete.

References

1 L. Aceto and A. Ingólfsdóttir. *Reactive Systems: Modelling, Specification and Verification.* Cambridge University Press, 2007.

2 J. L. Balcázar, J. Gabarró, and M. Santha. Deciding bisimilarity is P-complete. *Form. Asp. Comput.*, 4(6A):638–648, 1992.

3 D. A. M. Barrington and J. Corbet. On the relative complexity of some languages in NC^1. *Inform. Process. Lett.*, 32:251–256, 1989.

4 S. R. Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Kurt Gödel Colloquium 97*, pages 18–33, 1997.

5 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer, 2013.

6 M. Elberfeld, A. Jakoby, and T. Tantau. Logspace versions of the theorems of Bodlaender and Courcelle. In *Proceedings of FOCS 2010*, pages 143–152. IEEE, 2010.

7 M. Elberfeld, A. Jakoby, and T. Tantau. Algorithmic meta theorems for circuit classes of constant and logarithmic depth. In *Proceedings of STACS 2012*, volume 14 of *LIPIcs*, pages 66–77. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

8 M. Elberfeld and P. Schweitzer. Canonizing graphs of bounded tree width in logspace. In *Proceedings of STACS 2016*, volume 47 of *LIPIcs*, pages 32:1–32:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

9 R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, 1995.

10 M. Grohe, B. Grußien, A. Hernich, and B. Laubner. L-recursion and a new logic for logarithmic space. *Log. Meth. Comput. Sci.*, 9(1), 2012.

11 F. Gurski and E. Wanke. The tree-width of clique-width bounded graphs without $K_{n,n}$. In *Proceedings of WG 2000*, volume 1928 of *Lecture Notes in Computer Science*, pages 196–205. Springer, 2000.

12 W. Hesse, E. Allender, and D. A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65:695–716, 2002.

13 B. Jenner, J. Köbler, P. McKenzie, and J. Torán. Completeness results for graph isomorphism. *J. Comput. Syst. Sci.*, 66(3):549–566, 2003.

14 M. Kaminski, V. V. Lozin, and M. Milanic. Recent developments on graphs of bounded clique-width. *Discrete Appl. Math.*, 157(12):2747–2761, 2009.

15 P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Inform. Comput.*, 86(1):43–68, 1990.

16 K.-J. Lange and P. McKenzie. On the Complexity of Free Monoid Morphisms. In *Proceedings of ISAAC'98*, volume 1533 of *Lecture Notes in Computer Science*, pages 247–256. Springer, 1998.

17 S. Lindell. A logspace algorithm for tree canonization (extended abstract). In *Proceedings of STOC 1992*, pages 400–404. ACM, 1992.

18 J. Srba. On the power of labels in transition systems. In *Proceedings of CONCUR 2001*, volume 2154 of *Lecture Notes in Computer Science*, pages 277–291. Springer, 2001.

19 H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.

Monadic Second Order Finite Satisfiability and Unbounded Tree-Width*

Tomer Kotek[1], Helmut Veith[2], and Florian Zuleger[3]

1 **TU Vienna, Austria**
2 **TU Vienna, Austria**
3 **TU Vienna, Austria**

Abstract

The finite satisfiability problem of monadic second order logic is decidable only on classes of structures of bounded tree-width by the classic result of Seese [25]. We prove that the following problem is decidable:

> **Input:** (i) A monadic second order logic sentence α, and (ii) a sentence β in the two-variable fragment of first order logic extended with counting quantifiers. The vocabularies of α and β may intersect.
> **Output:** Is there a finite structure which satisfies $\alpha \wedge \beta$ such that the restriction of the structure to the vocabulary of α has bounded tree-width? (The tree-width of the desired structure is not bounded.)

As a consequence, we prove the decidability of the satisfiability problem by a finite structure of bounded tree-width of a logic $\mathrm{MSO}^{\exists \mathrm{card}}$ extending monadic second order logic with linear cardinality constraints of the form $|X_1| + \cdots + |X_r| < |Y_1| + \cdots + |Y_s|$ on the variables X_i, Y_j of the outer-most quantifier block. We prove the decidability of a similar extension of WS1S.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Monadic Second Order Logic MSO, Two variable Fragment with Counting C2, Finite decidability, Unbounded Tree-width, WS1S with Cardinality Constraints

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.13

1 Introduction

Monadic second order logic (MSO) is among the most expressive logics with good algorithmic properties. It has found countless applications in computer science in diverse areas ranging from verification and automata theory [13, 19, 26] to combinatorics [16, 18], and parameterized complexity theory [8, 6].

The power of MSO is most visible over graphs of bounded tree-width, and with second order quantifiers ranging over sets of edges[1]: (1) Courcelle's famous theorem shows that MSO model checking is decidable over graphs of bounded tree-width in linear time [5, 1]. (2) Finite satisfiability by graphs of bounded tree-width is decidable [5] (with non-elementary complexity) – thus contrasting Trakhtenbrot's undecidability result of first order logic. (3) Seese proved [25] that for each class $\mathcal{K}$ of graphs with *unbounded* tree-width, finite satisfiability of MSO by graphs in $\mathcal{K}$ is undecidable. Together, (2) and (3) give a fairly clear

* The tragic death of Helmut Veith prevented him from approving the final version. All faults and inaccuracies belong to his co-authors.
[1] The logic we denote by MSO is denoted MS_2 by Courcelle and Engelfriet [6].

© Tomer Kotek, Helmut Veith, and Florian Zuleger;
licensed under Creative Commons License CC-BY

picture of the decidability of finite satisfiability of MSO. It appeared that (3) gives a natural limit for decidability of MSO on graph classes. For instance, finite satisfiability on planar graphs is undecidable because their tree-width is unbounded.

While Courcelle and Seese circumvent Trakhtenbrot's undecidability result by *restricting the classes of graphs (or relational structures)*, several other research communities have studied syntactic restrictions of first order logic. Modal logic [27], many temporal logics [22], [24, Chapter 24], the guarded fragment [9], many description logics [2], and the two-variable fragment [10] are restricted first order logics with decidable finite satisfiability, and hundreds of papers on these topics have explored the border between decidability and undecidability. While many of the earlier papers exploited variations of the tree model property to show decidability, recent research has also focused on logics such as the two-variable fragment with counting C^2 [11, 23], where finite satisfiability is decidable despite the absence of the tree model property. In a recent breakthrough result, Charatonik and Witkowski [4] have extended this result to structures with built-in binary trees. Note that this logic is not a fragment of first order logic, but more naturally understood as a very weak second order logic which can express one specific second order property – the property of being a tree.

Our main result is a powerful generalization of the seminal result on decidability of the satisfiability problem of MSO over bounded tree-width and the recent theorem by [4]: We show decidability of finite satisfiability of conjunctions $\alpha \wedge \beta$ where α is in MSO and β is in C^2 by a finite structure $\mathfrak{M}$ whose restriction to the vocabulary of α has bounded tree-width. (Theorem 3.1 in Section 3)

Let us put this result into perspective:

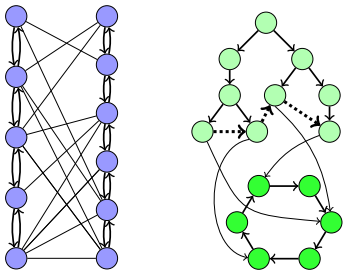
- The MSO decidability problem is a trivial consequence by setting β to true; Charatonik and Witkowski's result follows by choosing α to be an MSO formula which axiomatizes a d-ary tree, which is a standard construction [6].
- The decidability of *model checking* $\alpha \wedge \beta$ over a finite structure is a much simpler problem than ours: We just have to model check α and β one after the other. In contrast, satisfiability is not obvious because α and β can share relational variables. running two finite satisfiability algorithms for the two formulas independently may yield two models which disagree on the shared vocabulary. Thus, the problem we consider is similar in spirit to (but technically very different from) Nelson-Oppen [21] combinations of theories.
- Our result trivially generalizes to Boolean combinations of sentences in the two logics.

Proof Technique

We show how to reduce our satisfiability problem for $\alpha \wedge \beta$ to the finite satisfiability of a C^2-sentence with a built-in tree, which is decidable by [4]. The most significant technical challenge is to eliminate shared binary relation symbols between the two conjuncts. Our Separation Theorem overcomes this challenge by a construction based on local types of universe elements and a coloring argument for directed graphs. The second technical challenge is to replace the MSO-sentence α with an equi-satisfiable C^2-sentence α'. To do so, we apply tools including the Feferman-Vaught theorem for MSO and translation schemes.

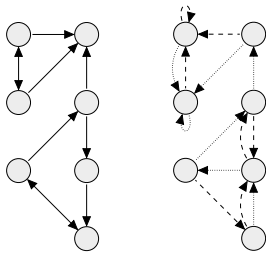
Monadic Second Order Logic with Cardinalities

Our main theorem implies new decidability results for monadic second order logic with cardinality constraints, i.e., expressions of the form $|X_1| + \ldots |X_r| < |Y_1| + \ldots |Y_t|$ where X_i and Y_i are monadic second order variables. Klaedtke and Rueß [15] showed that the decision problem for the theory of weak monadic second order logic with cardinality constraints of one



(a) (b)

Figure 1



(a) (b)

Figure 2

successor ($\mathrm{WS1S}^{\mathrm{card}}$) is undecidable; they described a decidable fragment where the second order quantifiers have no alternation and appear *after* the first order quantifiers in the prefix. Our main theorem implies decidability of a different fragment of WS1S with cardinalities: The fragment $\mathrm{MSO}^{\exists \mathrm{card}}$ consists of formulas $\exists \bar{X} \psi$ where the cardinality constraints in ψ involve only the monadic second order variables from $\bar{X}$, cf. Theorem 7.1 in Section 7. Note that in contrast to [15], our fragment is a strict superset of WS1S.

For WS2S, we are not aware of results about decidable fragments with cardinalities. We describe a strict superset of MSO whose satisfiability problem over finite graphs of bounded tree-width is decidable, and which is syntactically similar to the WS1S extension above.

Expressive Power over Structures

Our main result extends the existing body of results on finite satisfiability by structures of bounded tree-width to a significantly richer set of structures. The structures we consider are C^2-axiomatizable extensions of structures of bounded tree-width. For instance, we can have interconnected doubly-linked lists as in Fig. 1(a), or a tree whose leaves are connected in a chain and have edges pointing to any of the nodes of a cyclic list as in Fig. 1(b). Such structures occur very naturally as shapes of dynamic data structures in programming – where cycles and trees are containers for data, and additional edges express relational information between the data. The analysis of semantic relations between data structures served as a motivation for us to investigate the logics in the current paper [3].

Being a cyclic list or a tree whose leaves are chained can be expressed in MSO and both of these data structures have tree-width at most 3. We can compel the edges between the tree and the cyclic list to obey C^2-expressible constraints such as:

- every leaf of the tree has a single edge to the cyclic list;
- every node of the cyclic list has an incoming edge from at least one leaf of the tree; or
- any two leaves pointing to the same node of the cyclic list agree on membership in some unary relation.

Note that while the structures we consider may contain grids of unbounded sizes as subgraphs, the logic cannot axiomatize them.

2 Background

This section introduces basic definitions and results in model theory and graph theory. We follow [20] and [6].

The **two-variable fragment with counting** C^2 is the extension of the two-variable fragment of first order logic with first order counting quantifiers $\exists^{\leq n}$, $\exists^{\geq n}$, $\exists^{=n}$, for every $n \in \mathbb{N}$. Note that C^2 remains a fragment of first order logic. **Monadic Second order logic** MSO is the extension of first order logic with set quantifiers which can range over elements of the universe or subsets of relations[2]. Throughout the paper all structures consist of unary and binary relations only. Structures are finite unless explicitly stated otherwise (in the discussion of WS1S). Let $\mathcal{C}$ be a vocabulary (signature). The **arity** of a relation symbol $C \in \mathcal{C}$ is denoted by $arity(C)$. The set of unary (binary) relation symbols in $\mathcal{C}$ are $\mathbf{un(\mathcal{C})}$ $\mathbf{(bin(\mathcal{C}))}$. We write MSO($\mathcal{C}$) for the set of MSO-formulas on the vocabulary $\mathcal{C}$. The quantifier rank of a formula $\varphi \in$ MSO, i.e. the maximal depth of nested quantifiers in φ is denoted $\mathrm{qr}(\varphi)$. We denote by $\mathfrak{A}_1 \sqcup \mathfrak{A}_2$ the **disjoint union** of two $\mathcal{C}$-structures $\mathfrak{A}_1$ and $\mathfrak{A}_2$. Given vocabularies $\mathcal{C}_1 \subseteq \mathcal{C}_2$, a $\mathcal{C}_2$-structure $\mathfrak{A}_2$ is an **expansion** of a $\mathcal{C}_1$-structure $\mathfrak{A}_1$ if $\mathfrak{A}_1$ and $\mathfrak{A}_2$ agree on the symbols in $\mathcal{C}_1$; in this case $\mathfrak{A}_1$ is the **reduct** of $\mathfrak{A}_2$ to $\mathcal{C}_1$, i.e. $\mathfrak{A}_1$ is the $\mathcal{C}_1$-reduct of $\mathfrak{A}_2$. We denote the reduct of $\mathfrak{A}_2$ to $\mathcal{C}_1$ by $\mathfrak{A}_2|_{\mathcal{C}_1}$. A $\mathcal{C}$-structure $\mathfrak{A}_0$ with universe A_0 is a **substructure** of a $\mathcal{C}$-structure $\mathfrak{A}_1$ with universe A_1 if $A_0 \subseteq A_1$ and for every $R \in \mathcal{C}$, $R^{\mathfrak{A}_0} = R^{\mathfrak{A}_1} \cap A_0^{arity(R)}$. We say that $\mathfrak{A}_0$ is the substructure of $\mathfrak{A}_1$ generated by A_0.

Graphs are structures of the vocabulary[3] $\mathcal{C}_G = \langle s \rangle$ consisting of a single binary relation symbol s. Graphs are undirected without multiple edges but possibly with self-loops unless explicitly stated otherwise. **Tree-width** $\mathrm{tw}(G)$ is a graph parameter indicating how close a simple undirected graph G is to being a tree, cf. [6]. It is well-known that a graph has tree-width at most k iff it is a partial k-tree. A **partial k-tree** is a subgraph of a k-tree. **k-trees** are built inductively from the $(k+1)$-clique by repeated addition of vertices, each of which is connected with k edges to a k-clique. The **Gaifman graph** $\mathrm{Gaif}(\mathfrak{A})$ of a $\mathcal{C}$-structure $\mathfrak{A}$ is the graph whose vertex set is the universe of $\mathfrak{A}$ and whose edge set is the union of the symmetric closures of $C^{\mathfrak{A}}$ for every $C \in \mathrm{bin}(\mathcal{C})$. Note the unary relations of $\mathfrak{A}$ play no role in $\mathrm{Gaif}(\mathfrak{A})$. The **tree-width $\mathbf{tw(\mathfrak{A})}$ of a $\mathcal{C}$-structure** $\mathfrak{A}$ is the tree-width of its Gaifman graph. In this paper, tree-width is a parameter of *finite* structures only. Fix $k \in \mathbb{N}$ for the rest of the paper. k will denote the tree-width bound we consider.

We introduce the notion of **oriented k-trees** which refines the notion of k-trees. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be a vocabulary consisting of binary relation symbols. An oriented k-tree is an $\mathcal{R}$-structure $\mathfrak{R}$ in which all $R_i^{\mathfrak{R}}$ are total functions and whose Gaifman graph $\mathrm{Gaif}(\mathfrak{R})$ is a partial k-tree.

▶ **Lemma 2.1.** *Every $\mathcal{C}$-structure $\mathfrak{M}$ of tree-width k can be expanded into a $(\mathcal{C} \cup \mathcal{R})$-structure $\mathfrak{N}$ such that:*

(i) $\mathfrak{N}|_{\mathcal{R}}$ *is an oriented k-tree,*

(ii) $\mathrm{Gaif}(\mathfrak{N})$ *is a subgraph of* $\mathrm{Gaif}(\mathfrak{N}|_{\mathcal{R}})$*, and*

(iii) *the tree-width of* $\mathfrak{N}$ *is k.*

The oriented 2-tree in Fig. 2(b) is an expansion of the directed graph in Fig. 2(a) as guaranteed in Lemma 2.1. In Fig. 2(b), R_1 and R_2 are denoted by the dashed arrows and the dotted arrows, respectively. There are several other oriented k-trees which expand Fig. 2(a) and fulfill the requirements in Lemma 2.1,

To see that Lemma 2.1 holds, we describe a construction of $\mathfrak{N}$ echoing the process of constructing k-trees above. For each vertex u of the initial $(k+1)$-clique, we can set the

[2] On relational structures, MSO is also known as *Guarded Second Order logic* GSO. The results of this paper extend to CMSO, the extension of MSO with modular counting quantifiers.

[3] Since we explicitly allowed quantification over subsets of relations for MSO, we do not view graphs and structures as incidence structures, in contrast to [6, Sections 1.8.1 and 1.9.1].

values of $R_1^{\mathfrak{N}}(u), \ldots, R_k^{\mathfrak{N}}(u)$ to be the other k vertices of the clique. When a new vertex u is added to the k-tree, k edges incident to it are added. We set $R_1^{\mathfrak{N}}(u), \ldots, R_k^{\mathfrak{N}}(u)$ to be the set of vertices incident to u. For oriented k-trees whose Gaifman graph is not a k-tree the construction of an oriented k-tree is augmented by changing the value of $R_i^{\mathfrak{N}}(u)$ to $R_i^{\mathfrak{N}}(u) = u$ whenever $R_i^{\mathfrak{N}}(u)$ is not well-defined. This can happen when the target of u under $R_i^{\mathfrak{N}}$ is a vertex which was eliminated by taking the subgraph of a k-tree to obtain the partial k-tree.

3 Overview of the Main Theorem and its Proof

The precise statement of the main theorem is as follows:

▶ **Theorem 3.1** (Main Theorem). *Let $\mathcal{C}_{bnd}$ and $\mathcal{C}_{unb}$ be vocabularies. Let s be a binary relation symbol not in $\mathcal{C}_{bnd} \cup \mathcal{C}_{unb}$. Let $\alpha \in \mathrm{MSO}(\mathcal{C}_{bnd})$ and $\beta \in C^2(\mathcal{C}_{unb})$. There is an effectively computable sentence $\delta \in C^2(\mathcal{D})$ over a vocabulary $\mathcal{D} \supseteq \{s\}$ such that the following are equivalent:*

(i) *There is a $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structure $\mathfrak{M}$ such that $\mathfrak{M} \models \alpha \wedge \beta$ and $\mathrm{tw}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) \leq k$.*

(ii) *There is a $\mathcal{D}$-structure $\mathfrak{N}$ such that $\mathfrak{N} \models \delta$ and $s^{\mathfrak{N}}$ is a binary tree.*

The first step towards proving Theorem 3.1 is the Separation Theorem:

▶ **Theorem 3.2** (Separation Theorem). *Let $\mathcal{C}_{bnd}$ and $\mathcal{C}_{unb}$ be vocabularies. Let $\alpha \in \mathrm{MSO}(\mathcal{C}_{bnd})$ and $\beta \in C^2(\mathcal{C}_{unb})$. There are effectively computable sentences $\alpha' \in \mathrm{MSO}(\mathcal{D}_{bnd})$ and $\beta' \in C^2(\mathcal{D}_{unb})$ over vocabularies $\mathcal{D}_{bnd}$ and $\mathcal{D}_{unb}$ such that $\mathcal{D}_{bnd} \cap \mathcal{D}_{unb}$ only contains unary relation symbols and the following are equivalent:*

(i) *There is a $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structure $\mathfrak{M}$ with $\mathfrak{M} \models \alpha \wedge \beta$ and $\mathrm{tw}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) \leq k$*

(ii) *There is a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure $\mathfrak{N}$ with $\mathfrak{N} \models \alpha' \wedge \beta'$ and $\mathrm{tw}(\mathfrak{N}|_{\mathcal{D}_{bnd}}) \leq k$.*

In conjunction with Theorem 3.2, we only need to prove Theorem 3.1 in the case that the MSO-formula α and the C^2-formula β only share unary relation symbols. The significance of Theorem 3.2 is that it allows us to use tools designed for MSO in our more involved setting. The proof of Theorem 3.2 uses notions of types for C^2-sentences in Scott normal form, coloring arguments, and an induction on ranks of structures. Theorem 3.2 is discussed in Section 4. The next step is to move from structures whose reducts have bounded tree-width to structures which contain a binary tree.

▶ **Lemma 3.3.** *Let $\mathcal{C}_{bnd}$ and $\mathcal{C}_{unb}$ be vocabularies such that $\mathcal{C}_{bnd} \cap \mathcal{C}_{unb}$ contains only unary relation symbols. Let s be a binary relation symbol. There is a vocabulary $\mathcal{D}_{bnd}$ consisting of s and unary relation symbols only as follows. For every $\alpha \in \mathrm{MSO}(\mathcal{C}_{bnd})$ and $\beta \in C^2(\mathcal{C}_{unb})$, there are effectively computable sentences $\alpha' \in \mathrm{MSO}(\mathcal{D}_{bnd})$ and $\beta' \in C^2(\mathcal{D}_{bnd} \cup \mathcal{C}_{unb})$ such that the following are equivalent:*

(i) *There is a $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structure $\mathfrak{M}$ such that $\mathfrak{M} \models \alpha \wedge \beta$ and $\mathrm{tw}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) \leq k$.*

(ii) *There is a $(\mathcal{D}_{bnd} \cup \mathcal{C}_{unb})$-structure $\mathfrak{N}$ such that $\mathfrak{N} \models \alpha' \wedge \beta'$ and $s^{\mathfrak{N}}$ is a binary tree.*

Technically, Lemma 3.3 is proved using a translation scheme which maps structures with a binary tree into structures whose $\mathcal{C}_{bnd}$-reducts have tree-width at most k, and conversely, each of the latter structures is the image of a structure with a binary tree under the translation scheme. Translation schemes capturing the graphs of tree-width at most k as the image of labeled trees were studied in the context of decidability and model checking of MSO [1]. We need a more refined construction to ensure that the translation scheme also behaves correctly on C^2-sentences, i.e. that it maps C^2-sentences to C^2-sentences, see Lemma 3.3 in Section 5.

Now that we have reduced our attention to the case that our structures contain a binary tree, we can replace MSO-sentences with equi-satisfiable C^2-sentences.

▶ **Lemma 3.4.** *Let $\mathcal{C}$ be a vocabulary which consists only of a binary relation symbol s and unary relation symbols. Let α be an* MSO($\mathcal{C}$)*-sentence. There is an effectively computable $C^2(\mathcal{D})$-sentence γ over a vocabulary $\mathcal{D} \supseteq \mathcal{C}$ such that for every $\mathcal{C}$-structure $\mathfrak{M}$ in which $s^{\mathfrak{M}}$ is a binary tree the following are equivalent:*

(i) $\mathfrak{M} \models \alpha$.

(ii) *There is a $\mathcal{D}$-structure $\mathfrak{N}$ expanding $\mathfrak{M}$ such that $\mathfrak{N} \models \gamma$.*

For the proof of Lemma 3.4 we use a Feferman-Vaught type theorem which states that the Hintikka type (i.e. MSO types) of a binary tree labeled with unary relation symbols depends only on the Hintikka types of its children. We can therefore axiomatize in C^2 that the Hintikka type of the labeled binary tree implies a given MSO-sentence.

Having replaced the MSO-sentence in statement (ii) of Lemma 3.3 with a C^2-sentence, we are left with the problem of deciding whether a C^2-sentence is satisfiable by a structure in which a specified relation is a binary tree, which has recently been shown to be decidable:

▶ **Theorem 3.5** (Charatonik and Witkowski [4])**.** *Let $\mathcal{C}$ be a vocabulary which contains a binary relation symbol s. Given a $C^2(\mathcal{C})$-sentence φ, it is decidable whether φ is satisfiable by a structure $\mathfrak{M}$ in which $s^{\mathfrak{M}}$ is a binary tree.*

4 Separation Theorem

4.1 Basic Definitions and Results

1-types and 2-types

We begin with some notation and definitions in the spirit of the literature on decidability of C^2, cf. e.g. [23, 4]. Let $\mathcal{A}$ be a vocabulary of unary and binary relations.

A **1-type** π is a maximal consistent set of atomic $\mathcal{A}$-formulas or negations of atomic $\mathcal{A}$-formulas with free variable x, i.e., exactly one of $A(x)$ and $\neg A(x)$ belongs to π for every unary relation symbol $A \in \mathcal{A}$, and exactly one of $B(x,x)$ and $\neg B(x,x)$ belongs to π for every binary relation symbol $B \in \mathcal{A}$. We denote by $\mathfrak{char}_\pi(x) = \bigwedge_{\iota \in \pi} \iota$ the formula that characterizes the 1-type π. We denote by 1-Types($\mathcal{A}$) the set of 1-types over $\mathcal{A}$.

A **2-type** λ is a maximal consistent set of atomic $\mathcal{A}$-formulas or negations of atomic $\mathcal{A}$-formulas with free variables x and y and $x \not\approx y \in \lambda$, i.e., for every $z \in \{x, y\}$ and unary relation symbol $A \in \mathcal{A}$, exactly one of $A(z)$ and $\neg A(z)$ belongs to λ, and for every $z_1, z_2 \in \{x, y\}$ and binary relation symbol $B \in \mathcal{A}$, exactly one of $B(z_1, z_2)$ and $\neg B(z_1, z_2)$ belongs to λ. We write λ^{-1} for the 2-type obtained from λ by substituting all occurrences of x resp. y with y resp. x. We write λ_x for the 1-type obtained from λ by restricting λ to formulas with free variable x. We write λ_y for the 1-type obtained from λ by restricting λ to formulas with free variable y *and substituting y with x*. We denote by $\mathfrak{char}_\lambda(x,y) = \bigwedge_{\iota \in \lambda} \iota$ the formula that characterizes the 2-type λ. We denote by 2-Types($\mathcal{A}$) the set of 2-types over $\mathcal{A}$.

Let $\mathfrak{M}$ be an $\mathcal{A}$-structure. We denote by $\text{1-}\mathfrak{tp}^{\mathfrak{M}}(u)$ the unique 1-type π such that $\mathfrak{M} \models \mathfrak{char}_\pi(u)$. For elements u, v of $\mathfrak{M}$, we denote by $\text{2-}\mathfrak{tp}^{\mathfrak{M}}(u,v)$ the unique 2-type λ such that $\mathfrak{M} \models \mathfrak{char}_\lambda(u,v)$. We denote by $\text{2-}\mathfrak{tp}(\mathfrak{M}) = \{\text{2-}\mathfrak{tp}^{\mathfrak{M}}(u,v) \mid u, v \text{ elements of } \mathfrak{M}\}$ the set of 2-types *realized* by $\mathfrak{M}$. The following lemma is easy to see:

▶ **Lemma 4.1.** *Let $\mathfrak{M}_1, \mathfrak{M}_2$ be two $\mathcal{A}$-structures over the same universe M and let $\phi = \forall x, y.\, \chi \in C^2(\mathcal{A})$ with χ quantifier-free. If $\text{2-}\mathfrak{tp}(\mathfrak{M}_1) = \text{2-}\mathfrak{tp}(\mathfrak{M}_2)$, then $\mathfrak{M}_1 \models \phi$ iff $\mathfrak{M}_2 \models \phi$.*

Scott Normal Form and $\mathcal{T}$-functionality

C^2-sentences have a Scott-Normal Form, cf. [12], which can be obtained by iteratively applying Skolemization and introducing new predicates for subformulas, together with predicates ensuring the soundness of this transformation:

▶ **Lemma 4.2** (Scott Normal Form, [12]). *For every C^2-sentence β there is a C^2-sentence β' of the form*

$$\forall x, y.\, \chi \wedge \bigwedge_{i \in [l]} \forall x.\, \exists^{=1} y.\, S_i(x, y), \tag{1}$$

with χ quantifier-free, over an expanded vocabulary such that β and β' are equi-satisfiable. Moreover, β' is computable. The expanded vocabulary contains in particular the fresh binary relation symbols $\mathcal{S} = \{S_1, \dots, S_l\}$.

Let $\mathcal{T}$ be a set of binary relation symbols. We say a structure $\mathfrak{M}$ is **$\mathcal{T}$-functional**, if for every $T \in \mathcal{T}$, $T^{\mathfrak{M}}$ is a total function on the universe of $\mathfrak{M}$. Observe the following are equivalent for every structure $\mathfrak{M}$:

(i) $\mathfrak{M}$ satisfies Eq. (1), and
(ii) $\mathfrak{M} \models \forall x, y.\, \chi$ and $\mathfrak{M}$ is $\mathcal{S}$-functional.

Message Types and Chromaticity

Let $\mathcal{T} \subseteq \mathrm{bin}(\mathcal{A})$ be a subset of the binary relation symbols of $\mathcal{A}$. We write $\lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{A})$ and say λ is a **$\mathcal{T}$-message type**, if $\lambda \in 2\text{-Types}(\mathcal{A})$ and $T(x, y) \in \lambda$ for some $T \in \mathcal{T}$. Let $\mathfrak{M}$ be a $\mathcal{A}$-structure with universe M. We define $E = \{(u, v) \in M^2 \mid \text{there is a } T \in \mathcal{T} \text{ with } \mathfrak{M} \models T(u, v)\}$. The **$\mathcal{T}$-message-graph** is the directed graph $G = (M, F)$, where $F = \{(u, v) \in M^2 \mid u \neq v \text{ and } (u, v) \in E \circ E\}$, where $R \circ S = \{(a, b) \mid \text{ there is a } c \text{ with } (a, c) \in R \text{ and } (c, b) \in S\}$ denotes the usual *composition* of relations. We say $\mathfrak{M}$ is **$\mathcal{T}$-chromatic**, if $1\text{-}\mathfrak{tp}^{\mathfrak{M}}(u) \neq 1\text{-}\mathfrak{tp}^{\mathfrak{M}}(v)$ for all $(u, v) \in F$.

We note that if $\mathfrak{M}$ is $\mathcal{T}$-functional, then G has out-degree $deg^+(u) \leq |\mathcal{T}|^2$ for all $u \in M$. This allows us to prove Lemma 4.3 based on Lemma 4.4.

▶ **Lemma 4.3.** *There is a finite set of unary relations symbols $\mathrm{colors}(\mathcal{T})$ such that every $\mathcal{T}$-functional $\mathcal{A}$-structure can be expanded to a $\mathcal{T}$-chromatic $(\mathcal{A} \cup \mathrm{colors}(\mathcal{T}))$-structure.*

▶ **Lemma 4.4.** *Let $G = (V, E)$ be a directed graph with out-degree $deg^+(v) \leq k$ for all $v \in V$. Then, the underlying undirected graph has a proper $(2k + 1)$-coloring.*

4.2 Separation Theorem

Let $G = (V, E)$ be an (undirected) graph. We say G is **k-bounded**, if the edges of G can be oriented such that every node of G has out-degree less than k. We say a structure $\mathfrak{M}$ is k-bounded if its Gaifman graph is k-bounded. We note that the formulation of separation theorem below is slightly more general than the formulation in Section 3 because it is stated in terms of k-bounded graphs (graphs with tree-width k are clearly k-bounded, see the discussion on oriented k-trees in Section 2).

▶ **Theorem 3.2** (Separation Theorem). *Let k be a natural number. Let $\mathcal{C}_{bnd}$ and $\mathcal{C}_{unb}$ be vocabularies. Let $\alpha \in \mathrm{MSO}(\mathcal{C}_{bnd})$ and[4] $\beta \in C^2(\mathcal{C}_{unb})$. There are effectively computable*

[4] The Separation Theorem remains correct if we replace C^2 with any logic containing C^2 which is closed under conjunction.

sentences $\alpha' \in \mathrm{MSO}(\mathcal{D}_{bnd})$ *and* $\beta' \in C^2(\mathcal{D}_{unb})$ *over vocabularies* $\mathcal{D}_{bnd}$ *and* $\mathcal{D}_{unb}$ *such that* $\mathcal{D}_{bnd} \cap \mathcal{D}_{unb}$ *only contains unary relation symbols such that for every* k*-bounded graph* G *the following are equivalent:*

(i) *There is a* $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$*-structure* $\mathfrak{M}$ *with* $\mathfrak{M} \models \alpha \wedge \beta$ *and* $\mathrm{Gaif}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) = G$.

(ii) *There is a* $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$*-structure* $\mathfrak{N}$ *with* $\mathfrak{N} \models \alpha' \wedge \beta'$ *and* $\mathrm{Gaif}(\mathfrak{N}|_{\mathcal{D}_{bnd}}) = G$.

We assume that β is in the form given in Eq. (1) for some set of binary relation symbols $\mathcal{S} = \{S_1, \ldots, S_l\} \subseteq \mathcal{C}_{unb}$ and quantifier-free C^2-formula χ. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ be a set of fresh binary relation symbols. We set $\mathcal{T} = \mathcal{S} \cup \mathcal{R}$. We begin by giving an intuition for the proof of the Separation Theorem in three stages.

4.2.1 Syntactic separation coupled with semantic constraints

For a binary relation symbol B, we define its *copy* as the relation symbol $\overline{B}$. For every vocabulary $\mathcal{A}$, we define its *copy* $\overline{\mathcal{A}} = \mathrm{un}(\mathcal{A}) \cup \{\overline{B} \mid B \in \mathcal{A}\}$ to be the unary relation symbols of $\mathcal{A}$ plus the copies of its binary relations symbols. We assume that copied relation symbols are distinct from non-copied symbols, i.e., $\mathrm{bin}(\mathcal{A}) \cap \mathrm{bin}(\overline{\mathcal{A}}) = \emptyset$. For a formula φ over vocabulary $\mathcal{A}$, we define its *copy* $\overline{\varphi}$ over vocabulary $\overline{\mathcal{A}}$ as the formula obtained from φ by substituting every occurrence of a binary relation symbol $B \in \mathcal{A}$ with $\overline{B}$.

The sentences $\overline{\alpha}$ (the copy of α) and β do not share any binary relation symbols. Clearly, (i) from Theorem 3.2 holds iff

(I) $\overline{\alpha} \wedge \beta$ is satisfied by a $((\overline{\mathcal{C}_{bnd}} \cup \overline{\mathcal{C}_{unb}}) \cup (\mathcal{C}_{bnd} \cup \mathcal{C}_{unb}))$-structure $\mathfrak{N}$ with $B^{\mathfrak{N}} = \overline{B}^{\mathfrak{N}}$ for all $B \in \mathrm{bin}(\mathcal{C}_{bnd})$ and $\mathrm{Gaif}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) = G$.

In the next two stages we will construct α' and β' so that (I) is equivalent to (ii) from Theorem 3.2. More precisely, we will construct sentences $\mu_{bnd}, \mu_{unb} \in C^2(\mathcal{D}_{unb})$ with $\mathcal{D}_{unb} \supseteq \mathcal{C}_{bnd} \cup \mathcal{C}_{unb}$ and $\mathcal{D}_{bnd} = \overline{\mathcal{D}_{unb}}$ such that (I) is equivalent (II):

(II) $(\overline{\alpha} \wedge \overline{\mu_{bnd}}) \wedge (\beta \wedge \mu_{unb})$ is satisfied by a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure $\mathfrak{N}$ with $\mathrm{Gaif}(\mathfrak{N}|_{\mathcal{D}_{bnd}}) = G$.

4.2.2 Representation of k-bounded structures using functions and unary relations

k-bounded $\mathcal{A}$-structures $\mathfrak{A}$ can be represented by introducing new binary relation symbols interpreted as functions and new unary relation symbols as follows.

(a) We add k fresh relation symbols $\mathcal{R} = \{R_1, \ldots, R_k\}$ and axiomatize that these relations are interpreted as total functions.

(b) We add fresh unary relations P_λ for each $\mathcal{R}$-message type and axiomatize that every element labeled by P_λ has an outgoing edge with 2-type λ. The symbols P_λ are called *unary 2-type annotations*.

(c) We axiomatize that $\mathrm{Gaif}(\mathfrak{A}|_{\mathrm{bin}(\mathcal{A})}) = \mathrm{Gaif}(\mathfrak{A}|_{\mathcal{R}})$.

In other words, the functions interpreting $R_1, \ldots, R_k$ witness that $\mathfrak{A}$ can be oriented so that every node in the Gaifman graph of $\mathfrak{A}$ has outdegree at most k. The 2-type of each edge (u, v) in $\mathfrak{A}$ is encoded by putting the unary relation symbol P_λ of the 2-type of (u, v) on the source u in the orientation.

Theorem 3.2 as well as (I) and (II) involve reducts which are k-bounded structures. Given a $((\overline{\mathcal{C}_{bnd}} \cup \overline{\mathcal{C}_{unb}}) \cup (\mathcal{C}_{bnd} \cup \mathcal{C}_{unb}))$-structure $\mathfrak{N}$, we will use the above representation twice, on $\mathfrak{N}|_{\mathcal{C}_{bnd}}$ and $\mathfrak{N}|_{\overline{\mathcal{C}_{bnd}}}$, by axiomatizing that every element labeled by P_λ has an outgoing edge with 2-type λ and an outgoing edges with 2-type $\overline{\lambda}$. This allows us to replace the condition from (I) that $B^{\mathfrak{N}} = \overline{B}^{\mathfrak{N}}$ for all $B \in \mathrm{bin}(\mathcal{C}_{bnd})$ with the condition that $R_i^{\mathfrak{N}} = \overline{R_i}^{\mathfrak{N}}$ for all $R_i \in \mathcal{R}$.

4.2.3 Establishing the semantic condition of (I) by swapping edges

Here we discuss how to show the implication from (II) to (I). Let $\mathfrak{N}$ be a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure with $\mathfrak{N} \models (\overline{\alpha} \wedge \overline{\mu_{bnd}}) \wedge (\beta \wedge \mu_{unb})$. It simplifies the discussion to split a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure $\mathfrak{N}$ into two $\mathcal{D}_{unb}$-structures. The $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ is $\mathfrak{N}|_{\mathcal{D}_{unb}}$. The $\mathcal{D}_{unb}$-structure $\mathfrak{L}'$ is obtained from $\mathfrak{N}|_{\mathcal{D}_{bnd}}$ by renaming copies of relation symbols $\overline{B}$ to B — i.e., we define the $\mathcal{D}_{unb}$-structure $\mathfrak{L}'$ by setting $1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{N}}(u)$ for all $u \in M$ and setting $\mathfrak{L}' \models B(u,v)$ iff $\mathfrak{N} \models \overline{B}(u,v)$ for all $u, v \in M$ and $B \in \mathrm{bin}(\mathcal{D}_{bnd})$. The interpretations of the relations R_i might differ in $\mathfrak{L}$ and $\mathfrak{L}'$. Observe that we have $\mathfrak{L}' \models \alpha$ and $\mathfrak{L} \models \beta$. The *key idea of the proof* is to show the existence of a sequence of structures $\mathfrak{L} = \mathfrak{L}_0, \ldots, \mathfrak{L}_p$, where each $\mathfrak{N}_{i+1}$ is obtained from $\mathfrak{N}_i$ by *swapping edges*, until the interpretations of the relations R_i agree in $\mathfrak{L}_p$ and $\mathfrak{L}'$. The edge swapping operation is a local operation which involves changing the 2-types of at most 4 edges.

The edge swapping operation satisfies two crucial preservation requirements: edge swapping preserves (PR-1) the truth value of β, i.e. $\mathfrak{L}_p \models \beta$, and (PR-2) $\mathcal{R}$-functionality. The universal constraint $\forall x, y.\, \chi$ in β is maintained under edge swapping because this operation does not change the set of 2-types (see Lemma 4.1). To satisfy the preservation requirements (PR-1) and (PR-2), all that remains is to guarantee the existence of a sequence of edge swapping preserving $\mathcal{S}$-functionality and $\mathcal{R}$-functionality (which amounts to $\mathcal{T}$-functionality because of $\mathcal{T} = \mathcal{S} \cup \mathcal{R}$). We use two main techniques that guarantee the existence of edges for which edge swapping preserves $\mathcal{T}$-functionality: chromaticity and unary 2-type annotations. We will axiomatize that the structures $\mathfrak{L}$ and $\mathfrak{L}'$ are chromatic; we will take care that chromaticity is maintained during edge swaps. We will add fresh unary relation symbols P_λ for every $\mathcal{T}$-message type λ and axiomatize that every element of $\mathfrak{N}$ labeled by P_λ has an outgoing edge with 2-type λ and an outgoing edge with 2-type $\overline{\lambda}$; we will take care that such outgoing edges are maintained during edge swaps.

4.2.4 Proof of the Separation Theorem

We now start the formal proof of the Separation Theorem. Let $\mathrm{colors}(\mathcal{T})$ be the vocabulary from Lemma 4.3. We set $\mathcal{E} = \mathcal{C}_{bnd} \cup \mathcal{C}_{unb} \cup \mathcal{R} \cup \mathrm{colors}(\mathcal{T})$ and $\mathcal{P} = \{P_\lambda \mid \lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{E})\}$. We set $\mathcal{D}_{unb} = \mathcal{E} \cup \mathcal{P}$ and $\mathcal{D}_{bnd} = \overline{\mathcal{D}_{unb}}$. Next we will define formulas $\alpha^\circ \in \mathrm{MSO}(\mathcal{D}_{unb})$ and $\beta' \in C^2(\mathcal{D}_{unb})$, and set $\alpha' = \overline{\alpha^\circ} \in \mathrm{MSO}(\mathcal{D}_{bnd})$. We set

$\mu_{bnd} = \psi_{functionality} \wedge \psi_{gaifmann} \wedge \psi_{typesBnd} \wedge \psi_{chromaticity} \wedge \psi_{subgraph}$, $\alpha^\circ = \alpha \wedge \mu_{bnd}$,
$\mu_{unb} = \psi_{functionality} \wedge \psi_{gaifmann} \wedge \psi_{typesUnb} \wedge \psi_{chromaticity}$ and $\beta' = \beta \wedge \mu_{unb}$, where:

- $\psi_{functionality} = \bigwedge_{i \in [k]} \forall x.\, \exists^{=1} y.\, R_i(x,y)$
  The formula $\psi_{functionality}$ expresses that each R_i is interpreted as a total function.
- $\psi_{gaifmann} = \forall x, y.\, x \not\approx y \to \left(\bigvee_{i \in [k]} R_i(x,y) \vee R_i(y,x) \leftrightarrow \bigvee_{B \in \mathrm{bin}(\mathcal{C}_{bnd})} B(x,y) \vee B(y,x) \right)$
  The formula $\psi_{gaifmann}$ expresses that for every $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ with $\mathfrak{L} \models \psi_{gaifmann}$ we have that $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{R}}) = \mathrm{Gaif}(\mathfrak{L}|_{\mathcal{C}_{bnd}})$.
- $\psi_{subgraph} = \bigwedge_{B \in \mathrm{bin}(\mathcal{C}_{unb})} \forall x, y.\, x \not\approx y \to \left(B(x,y) \to \bigvee_{i \in [k]} R_i(x,y) \vee R_i(y,x) \right)$
  The formula $\psi_{subgraph}$ expresses that for every $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ with $\mathfrak{L} \models \psi_{subgraph}$ we have that $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{C}_{unb}})$ is a subgraph of $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{R}})$.
- $\psi_{typesUnb} = \bigwedge_{\lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{E})} \forall x.\, P_\lambda(x) \leftrightarrow \exists y. \mathfrak{char}_\lambda(x,y)$
  The formula $\psi_{typesUnb}$ expresses that for every $\mathcal{T}$-message type λ a node u satisfies the predicate P_λ iff u has an outgoing edge with 2-type λ.

- $\psi_{typesBnd} = \bigwedge_{\substack{\lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{E}), i \in [k]: \\ R_i(x,y) \in \lambda \text{ or } R_i(y,x) \in \lambda}} \forall x.\, P_\lambda(x) \leftrightarrow \exists y. \mathfrak{char}_\lambda(x,y)$

  The formula $\psi_{typesBnd}$ expresses that for every $\mathcal{T}$-message type λ, where λ contains the predicate $R_i(x,y)$ or $R_i(y,x)$ for some $i \in [k]$, a node u satisfies the predicate P_λ iff u has an outgoing edge with 2-type λ.
- $\psi_{chromaticity} = \bigwedge_{\substack{\lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{E}), T \in \mathcal{T}: \\ \neg T(y,x) \in \lambda}} \forall x, y.\, P_\lambda(x) \wedge T(y,x) \rightarrow \neg \mathfrak{char}_{\lambda_y}(y)$

  The formula $\psi_{chromaticity}$ expresses that for every $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ with $\mathfrak{L} \models \psi_{typesUnb}$, $\mathfrak{L}$ is chromatic iff $\mathfrak{L} \models \psi_{chromaticity}$.

The direction "(i) implies (ii)" of the Separation Theorem is straightforward to show by appropriately expanding the model of (i) to a model of (ii):

▶ **Lemma 4.5.** *Let G be a k-bounded graph. Let $\mathfrak{M}$ be a $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structure with $\mathfrak{M} \models \alpha \wedge \beta$ and $\mathrm{Gaif}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) = G$. $\mathfrak{M}$ can be expanded to a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure $\mathfrak{N}$ with $\mathfrak{N} \models \alpha' \wedge \beta'$ and $\mathrm{Gaif}(\mathfrak{N}|_{\mathcal{D}_{bnd}}) = G$.*

Proof. Because of $\mathrm{Gaif}(\mathfrak{M}|_{\mathcal{C}_{bnd}}) = G$ and G is k-bounded, we can expand $\mathfrak{M}$ to a $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb} \cup \mathcal{R})$-structure $\widehat{\mathfrak{M}}$ such that $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{R}}) = \mathrm{Gaif}(\mathfrak{L}|_{\mathcal{C}_{bnd}})$ and $R_i^{\widehat{\mathfrak{M}}}$ is a total function for all $i \in [k]$ (possibly adding self-loops for the relations R_i). Thus, $\widehat{\mathfrak{M}} \models \psi_{functionality} \wedge \psi_{gaifmann}$. According to Lemma 4.3, $\widehat{\mathfrak{M}}$ can be expanded to a chromatic structure $\widetilde{\mathfrak{M}}$ over vocabulary $\mathcal{E}$ with $\widetilde{\mathfrak{M}} \models \psi_{chromaticity}$. We expand $\widetilde{\mathfrak{M}}$ to a $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ such that for all $u \in M$ and $\lambda \in \mathcal{T}$-MsgTypes$(\mathcal{E})$ we have $\mathfrak{L} \models P_\lambda(u)$ iff there is an element v of $\mathfrak{L}$ such that $2\text{-}\mathfrak{tp}^{\mathfrak{L}|_\mathcal{E}}(u,v) = \lambda$. This definition gives us $\mathfrak{L} \models \psi_{typesUnb}$, and thus $\mathfrak{L} \models \beta'$. We expand $\mathfrak{L}$ to a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure $\mathfrak{N}$ such that for all $u, v \in M$ and $B \in \mathrm{bin}(\mathcal{D}_{unb})$ we have $\mathfrak{N} \models \overline{B}(u,v)$ iff $\mathfrak{N} \models B(u,v)$ and $\mathfrak{N} \models R_i(u,v)$ or $\mathfrak{N} \models R_i(v,u)$ for some $i \in [k]$. We note that $\mathfrak{N} \models \alpha'$. ◀

Now we turn to the direction "(ii) implies (i)". Let G be a k-bounded graph. Let $\mathfrak{N}$ be a $(\mathcal{D}_{bnd} \cup \mathcal{D}_{unb})$-structure with $\mathfrak{N} \models \alpha' \wedge \beta'$ and $\mathrm{Gaif}(\mathfrak{N}|_{\mathcal{D}_{bnd}}) = G$. Let M be the universe of $\mathfrak{N}$. We define the $\mathcal{D}_{unb}$-structure $\mathfrak{L}'$ by setting $1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{N}}(u)$ for all $u \in M$ and setting $\mathfrak{L}' \models B(u,v)$ iff $\mathfrak{N} \models \overline{B}(u,v)$ for all $u, v \in M$ and $B \in \mathrm{bin}(\mathcal{D}_{bnd})$. We note that $\mathfrak{L}' \models \alpha^\circ$ and $\mathrm{Gaif}(\mathfrak{L}') = G$. We define the $\mathcal{D}_{unb}$-structure $\mathfrak{L}$ by setting $\mathfrak{L} = \mathfrak{N}|_{\mathcal{D}_{unb}}$. We note that $\mathfrak{L} \models \beta'$.

We make the following definition: For $u \in M$ and $i \in [k]$ we set $rank_u^i(\mathfrak{L}, \mathfrak{L}') = 1$, if there are $v, w \in M$ with $\mathfrak{L} \models R_i(u,v)$, $\mathfrak{L}' \models R_i(u,w)$ and $v \neq w$; we set $rank_u^i(\mathfrak{L}, \mathfrak{L}') = 0$, otherwise. We set $rank_u(\mathfrak{L}, \mathfrak{L}') = \sum_{i \in [k]} rank_u^i(\mathfrak{L}, \mathfrak{L}')$ and $rank(\mathfrak{L}, \mathfrak{L}') = \sum_{u \in M} rank_u(\mathfrak{L}, \mathfrak{L}')$. *rank* measures the deviation of the relations $\mathcal{R}$ in $\mathfrak{L}$ and $\mathfrak{L}'$ (we note that there always are unique $v, w \in M$ for $u \in M$ with $\mathfrak{L} \models R_i(u,v)$, $\mathfrak{L}' \models R_i(u,w)$ because of $\mathfrak{L} \models \psi_{functionality}$ and $\mathfrak{L}' \models \psi_{functionality}$). *rank* has the following important property that relates the relations R_i with the 2-types of $\mathfrak{L}$ and $\mathfrak{L}'$:

▶ **Lemma 4.6.** *Let $u \in M$ be an element with $rank_u^i(\mathfrak{L}, \mathfrak{L}') = 0$ and let $\lambda \in$ 2-Types$(\mathcal{E})$ with $R_i \in \lambda$ for some $i \in [k]$. For all $v \in M$ we have $2\text{-}\mathfrak{tp}^{\mathfrak{L}|_\mathcal{E}}(u,v) = \lambda$ iff $2\text{-}\mathfrak{tp}^{\mathfrak{L}'|_\mathcal{E}}(u,v) = \lambda$.*

Proof. Let $v \in M$ be an element with $2\text{-}\mathfrak{tp}^{\mathfrak{L}|_\mathcal{E}}(u,v) = \lambda$. We get $\mathfrak{L} \models P_\lambda(u)$ because of $\mathfrak{L} \models \psi_{typesUnb}$. Because of $1\text{-}\mathfrak{tp}^{\mathfrak{L}}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(u)$ we have $\mathfrak{L}' \models P_\lambda(u)$. Because of $\mathfrak{L}' \models \psi_{typesBnd}$ we have $\lambda = 2\text{-}\mathfrak{tp}^{\mathfrak{L}'|_\mathcal{E}}(u,w)$ for some $w \in M$. $rank_u^i(\mathfrak{L}, \mathfrak{L}') = 0$ implies that $\mathfrak{L}' \models R_i(u,v)$. Because of $\mathfrak{L}' \models \psi_{functionality}$ we get $v = w$. In the same way one can show that $2\text{-}\mathfrak{tp}^{\mathfrak{L}'|_\mathcal{E}}(u,v) = \lambda$ implies $2\text{-}\mathfrak{tp}^{\mathfrak{L}|_\mathcal{E}}(u,v) = \lambda$. ◀

If there is no deviation of the relations $\mathcal{R}$ in $\mathfrak{L}$ and $\mathfrak{L}'$, i.e., if $rank(\mathfrak{L}, \mathfrak{L}') = 0$, then we have established the direction "(ii) implies (i)":

▶ **Lemma 4.7.** *If* $rank(\mathfrak{L}, \mathfrak{L}') = 0$*, then* $\mathfrak{L}|_{\mathcal{C}_{bnd} \cup \mathcal{C}_{unb}} \models \alpha \wedge \beta$ *and* $\mathrm{Gaif}(\mathfrak{L}_{\mathcal{C}_{bnd}}) = G$*.*

Proof. We show $\mathfrak{L}|_{\mathcal{C}_{bnd}} = \mathfrak{L}'|_{\mathcal{C}_{bnd}}$. Let $u, v \in M$ with $\mathfrak{L} \models B(u, v)$ for some $B \in \mathrm{bin}(\mathcal{C}_{bnd})$. If $u = v$ we get $\mathfrak{L}' \models B(u, v)$ because u has the same 1-type in $\mathfrak{L}$ and $\mathfrak{L}'$. We assume $u \neq v$ in the following. Because of $\mathfrak{L} \models \psi_{gaifmann}$ we have $\mathfrak{L} \models R_i(u, v)$ or $\mathfrak{L} \models R_i(v, u)$ for some $i \in [k]$. Because of $rank(\mathfrak{L}, \mathfrak{L}') = 0$ we have $rank^i_u(\mathfrak{L}, \mathfrak{L}') = 0$. By Lemma 4.6 we have $2\text{-}\mathfrak{tp}^{\mathfrak{L}|\mathcal{E}}(u, v) = 2\text{-}\mathfrak{tp}^{\mathfrak{L}'|\mathcal{E}}(u, v)$. Thus, $\mathfrak{L}' \models B(u, v)$. In the same way one can show that $\mathfrak{L}' \models B(u, v)$ implies $\mathfrak{L} \models B(u, v)$. Thus, $\mathfrak{L}|_{\mathcal{C}_{bnd}} = \mathfrak{L}'|_{\mathcal{C}_{bnd}}$.

We show $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{C}_{bnd}}) = \mathrm{Gaif}(\mathfrak{L}')$: We have that $\mathrm{Gaif}(\mathfrak{L}') = \mathrm{Gaif}(\mathfrak{L}'|_{\mathcal{R}})$ because of $\mathfrak{L}' \models \psi_{gaifmann}$ and $\mathfrak{L}' \models \psi_{subgraph}$. We have $\mathrm{Gaif}(\mathfrak{L}|_{\mathcal{C}_{bnd}}) = \mathrm{Gaif}(\mathfrak{L}|_{\mathcal{R}})$ because of $\mathfrak{L} \models \psi_{gaifmann}$. We have $\mathrm{Gaif}(\mathfrak{L}'|_{\mathcal{R}}) = \mathrm{Gaif}(\mathfrak{L}|_{\mathcal{R}})$ because of $rank(\mathfrak{L}, \mathfrak{L}') = 0$. Thus, the claim follows. ◀

Finally, we show that there always is a sequence of models $\mathfrak{L}_0, \ldots, \mathfrak{L}_p$ with $\mathfrak{L} = \mathfrak{L}_0$, $\mathfrak{L}_i \models \alpha^\circ$ for each $0 \leq i \leq p$ and $rank(\mathfrak{L}_p, \mathfrak{L}') = 0$. We obtain each $\mathfrak{L}_{i+1}$ from $\mathfrak{L}_i$ by swapping edges, which has the consequence that the universe does not change, every each node u keeps its 1-type and the set of realized 2-types remains invariant throughout the construction of the sequence.

▶ **Lemma 4.8.** *There is a sequence of* $\mathcal{D}_{unb}$*-structures* $\mathfrak{L}_0, \ldots, \mathfrak{L}_p$*, with universe* M *and* $\mathfrak{L} = \mathfrak{L}_0$*, such that:*

(1) $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(u)$ *for all* $u \in M$*,*
(2) $2\text{-}\mathfrak{tp}(\mathfrak{L}_i|_{\mathcal{E}}) = 2\text{-}\mathfrak{tp}(\mathfrak{L}_{i+1}|_{\mathcal{E}})$ *for all* $0 \leq i < p$*,*
(3) $\mathfrak{L}_i \models \alpha^\circ$*, (in particular* $\mathfrak{L}_i$ *is* $\mathcal{T}$*-functional and chromatic),*
(4) $rank(\mathfrak{L}_i, \mathfrak{L}') > rank(\mathfrak{L}_{i+1}, \mathfrak{L}')$ *for all* $0 \leq i < p$*, and* $rank(\mathfrak{L}_p, \mathfrak{L}') = 0$*.*

Proof. Assume we have already defined $\mathfrak{L}_i$ and $rank(\mathfrak{L}_i, \mathfrak{L}') > 0$. In the following we will define $\mathfrak{L}_{i+1}$. Because of $rank(\mathfrak{L}_i, \mathfrak{L}') > 0$ we can choose some elements $u, v, w \in M$ and $j \in [k]$ such that $\mathfrak{L}_i \models R_j(u, v)$, $\mathfrak{L}' \models R_j(u, w)$ and $v \neq w$. Let $\lambda = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u, v)$. We have $\lambda \in \mathcal{T}\text{-MsgTypes}(\mathcal{E})$ because of $R_j \in \mathcal{R}$. We have $\mathfrak{L}_i \models P_\lambda(u)$ because of $\mathfrak{L}_i \models \psi_{typesUnb}$. Because $1\text{-}\mathfrak{tp}(u)^{\mathfrak{L}_i} = 1\text{-}\mathfrak{tp}(u)^{\mathfrak{L}'}$ we have $\mathfrak{L}' \models P_\lambda(u)$. Thus, $\mathfrak{L}_i \models \mathfrak{char}_{\lambda_y}(v)$ and $\mathfrak{L}' \models \mathfrak{char}_{\lambda_y}(w)$. With $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(w) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(w)$ we get $\mathfrak{L}_i \models \mathfrak{char}_{\lambda_y}(w)$, and thus $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$. We proceed by a case distinction:

Case 1: λ^{-1} is a $\mathcal{T}$-message type

We have $\mathfrak{L}' \models P_{\lambda^{-1}}(w)$ because of $\mathfrak{L}' \models \psi_{typesBnd}$. We get $\mathfrak{L}_i \models P_{\lambda^{-1}}(w)$ because of $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(w) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(w)$. With $\mathfrak{L}_i \models \psi_{typesUnb}$, there is an element $a \in M$ such that $\lambda = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a, w)$ and $\mathfrak{L}_i \models P_\lambda(a)$. We note that $u \neq a$ because of $\mathfrak{L}_i \models R_j(a, w)$, $v \neq w$ and v is the unique element with $\mathfrak{L}_i \models R_j(u, v)$ (using $\mathfrak{L}_i \models \psi_{functionality}$). Moreover, $\mathfrak{L}_i \models \mathfrak{char}_{(\lambda^{-1})_y}(a)$ and $\mathfrak{L}' \models \mathfrak{char}_{(\lambda^{-1})_y}(u)$. With $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(u)$ we get $\mathfrak{L}_i \models \mathfrak{char}_{(\lambda^{-1})_y}(u)$, and thus $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a)$.

We note that the edges (u, w), (w, u), (a, v) and (v, a) do not have $\mathcal{T}$-message types (*) because $\mathfrak{L}_i$ is chromatic, $u \neq a$, $v \neq w$, $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$, $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a)$ and the edges (u, v), (v, u), (a, w) and (w, a) have $\mathcal{T}$-message types. Similarly, we get $\lambda \neq 2\text{-}\mathfrak{tp}^{\mathfrak{L}'|\mathcal{E}}(u, v)$ and $\lambda \neq 2\text{-}\mathfrak{tp}^{\mathfrak{L}'|\mathcal{E}}(a, w)$ (**) because $\mathfrak{L}'$ is chromatic, $\lambda = 2\text{-}\mathfrak{tp}^{\mathfrak{L}'|\mathcal{E}}(u, w)$ and λ and λ^{-1} are $\mathcal{T}$-message types.

We define $\mathfrak{L}_{i+1}$ as follows: The unary relations of $\mathfrak{L}_{i+1}$ are defined such that we have $1\text{-}\mathfrak{tp}(u)^{\mathfrak{L}_{i+1}} = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(u)$ for all elements $u \in M$. We obtain the binary relations of $\mathfrak{L}_{i+1}$ by

swapping the edges (u,w) and (u,v) as well as (a,w) and (a,v) in $\mathfrak{L}_i$: we set $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(u,w) = \lambda$, $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(a,v) = \lambda$, $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(a,w) = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a,v)$ and $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(u,v) = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u,w)$; these 2-types are well-defined because of $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$. All other 2-types in $\mathfrak{L}_{i+1}|_{\mathcal{E}}$ are the same as in $\mathfrak{L}_i|_{\mathcal{E}}$. This completes the definition of $\mathfrak{L}_{i+1}$.

We now argue that $\mathfrak{L}_{i+1}$ satisfies properties (1)-(4). Clearly, $\mathfrak{L}_{i+1}$ satisfies (1) by definition. Because we only swapped edges from $\mathfrak{L}_i$ to $\mathfrak{L}_{i+1}$ we have (2). Because we only swapped swapped edges from $\mathfrak{L}_i$ to $\mathfrak{L}_{i+1}$ we get $\mathfrak{L}_{i+1} \models \psi_{gaifmann}$ and $\mathfrak{L}_{i+1} \models \psi_{subgraph}$ from $\mathfrak{L}_i \models \psi_{gaifmann}$ and $\mathfrak{L}_i \models \psi_{subgraph}$. Because of (*) and $2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u,v) = \lambda = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a,w)$ we get $\mathfrak{L}_{i+1} \models \psi_{typesUnb}$ from $\mathfrak{L}_i \models \psi_{typesUnb}$, that $\mathfrak{L}_{i+1}$ is $\mathcal{T}$-functional because $\mathfrak{L}_i$ is $\mathcal{T}$-functional and $\mathfrak{L}_{i+1} \models \psi_{chromaticity}$ from $\mathfrak{L}_i \models \psi_{chromaticity}$ (using that $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$ and $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(a)$). With $\mathfrak{L}_i \models \alpha^\circ$ and Lemma 4.1 we get $\mathfrak{L}_{i+1} \models \alpha^\circ$. Thus, $\mathfrak{L}_{i+1}$ satisfies property (3).

It remains to show property (4). Using (*), (**) and Lemma 4.6 we get $rank_u(\mathfrak{L}_{i+1}, \mathfrak{L}') < rank_u(\mathfrak{L}_i, \mathfrak{L}')$ and $rank_z(\mathfrak{L}_{i+1}, \mathfrak{L}') \leq rank_z(\mathfrak{L}_i, \mathfrak{L}')$ for $z \in \{v, w, a\}$. Moreover, $rank_z(\mathfrak{L}_{i+1}, \mathfrak{L}') = rank_z(\mathfrak{L}_i, \mathfrak{L}')$ for $z \in M \setminus \{u, v, w, a\}$. Thus, $rank(\mathfrak{L}_i, \mathfrak{L}') > rank(\mathfrak{L}_{i+1}, \mathfrak{L}')$.

Case 2: λ^{-1} is not a $\mathcal{T}$-message type

We note that the edge (w,u) does not have a $\mathcal{T}$-message type because $\mathfrak{L}_i$ is chromatic, $v \neq w$, $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$ and the edge (u,v) has a $\mathcal{T}$-message type.

We define $\mathfrak{L}_{i+1}$ as follows: The unary relations of $\mathfrak{L}_{i+1}$ are defined such that we have $1\text{-}\mathfrak{tp}(u)^{\mathfrak{L}_{i+1}} = 1\text{-}\mathfrak{tp}(u)^{\mathfrak{L}_i}$ for all elements $u \in M$. We obtain the binary relations of $\mathfrak{L}_{i+1}$ by swapping edges in $\mathfrak{L}_i$: $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(u,w) = \lambda$, and $2\text{-}\mathfrak{tp}^{\mathfrak{L}_{i+1}|\mathcal{E}}(u,v) = 2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u,w)$. These 2-types are well-defined because of $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w)$. All other 2-types in $\mathfrak{L}_{i+1}|_{\mathcal{E}}$ are the same as in $\mathfrak{L}_i|_{\mathcal{E}}$. This completes the definition of $\mathfrak{L}_{i+1}$.

We now argue that $\mathfrak{L}_{i+1}$ satisfies properties (1)-(4). As in the previous case, one can argue that $\mathfrak{L}_{i+1}$ satisfies (1) and (2), $\mathfrak{L}_{i+1} \models \psi_{gaifmann}$ and $\mathfrak{L}_{i+1} \models \psi_{subgraph}$. Because (v,u) and (w,u) do not have $\mathcal{T}$-message types we get $\mathfrak{L}_{i+1} \models \psi_{typesUnb}$ from $\mathfrak{L}_i \models \psi_{typesUnb}$ and that $\mathfrak{L}_{i+1}$ is $\mathcal{T}$-functional because $\mathfrak{L}_i$ is $\mathcal{T}$-functional. We get $\mathfrak{L}_{i+1} \models \psi_{chromaticity}$ from $\mathfrak{L}_i \models \psi_{chromaticity}$, $\mathfrak{L}' \models \psi_{chromaticity}$ and $1\text{-}\mathfrak{tp}^{\mathfrak{L}_i}(w) = 1\text{-}\mathfrak{tp}^{\mathfrak{L}'}(w)$. With $\mathfrak{L}_i \models \alpha^\circ$ and Lemma 4.1 we get $\mathfrak{L}_{i+1} \models \alpha^\circ$. Thus, $\mathfrak{L}_{i+1}$ satisfies property (3).

It remains to show property (4). From Lemma 4.6 we get $2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(u,w) \neq \lambda$ and $2\text{-}\mathfrak{tp}^{\mathfrak{L}'|\mathcal{E}}(u,v) \neq \lambda$. Again applying Lemma 4.6 we get $rank_u(\mathfrak{L}_{i+1}, \mathfrak{L}') < rank_u(\mathfrak{L}_i, \mathfrak{L}')$. Because $2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(v,u)$ and $2\text{-}\mathfrak{tp}^{\mathfrak{L}_i|\mathcal{E}}(w,u)$ are not $\mathcal{T}$-message types, we get $rank_v(\mathfrak{L}_{i+1}, \mathfrak{L}') = rank_v(\mathfrak{L}_i, \mathfrak{L}')$ and $rank_w(\mathfrak{L}_{i+1}, \mathfrak{L}') = rank_w(\mathfrak{L}_i, \mathfrak{L}')$. Moreover, we have that $rank_z(\mathfrak{L}_{i+1}, \mathfrak{L}') = rank_z(\mathfrak{L}_i, \mathfrak{L}')$ for all $z \in M \setminus \{u, v, w\}$. Thus, $rank(\mathfrak{L}_i, \mathfrak{L}') > rank(\mathfrak{L}_{i+1}, \mathfrak{L}')$. ◀

5 From Bounded Tree-width to Binary Trees

This section is devoted to a discussion of the proof of Lemma 3.3. It is well-known that graphs of tree-width k can be encoded as trees whose vertices are labeled with a finite number of labels. The popular definition of tree-width based on tree decompositions can be used as the basis of such an encoding. Moreover, the class of graphs of tree-width k can be obtained as the image of a translation scheme on the class of trees [1]. A translation scheme is a tuple of formulas which induces an operation mapping structures to structures. Our proof of Lemma 3.3 relies on an encoding of structures of bounded tree-width into trees in terms of a translation scheme with some special properties.

A **translation scheme for $\mathcal{C}_2$ over $\mathcal{C}_1$** is a tuple $\mathbf{t} = \langle \phi, \psi_C : C \in \mathcal{C}_2 \rangle$ of MSO($\mathcal{C}_1$)-formulas such that ϕ has exactly one free first order variable and the number of free first order variables in each ψ_C is $arity(C)$. The formulas ϕ and ψ_C, $C \in \mathcal{C}_2$, do not have any free second order variables.[5] The **quantifier rank qr(t) of t** is the maximum of the quantifier ranks of ϕ and the ψ_C. $\mathbf{t}$ is **quantifier-free** if $\mathrm{qr}(\mathbf{t}) = 0$. The **induced transduction $\mathbf{t}^\star$** is a partial function from $\mathcal{C}_1$-structures to $\mathcal{C}_2$-structures which assigns a $\mathcal{C}_2$-structure $\mathbf{t}^\star(\mathfrak{A})$ to a $\mathcal{C}_1$-structure $\mathfrak{A}$ as follows. The universe of $\mathbf{t}^\star(\mathfrak{A})$ is $A_{\mathbf{t}} = \{a \in A : \mathfrak{A} \models \phi(a)\}$. The interpretation of $C \in \mathcal{C}_2$ in $\mathbf{t}^\star(\mathfrak{A})$ is $C^{\mathbf{t}^\star(\mathfrak{A})} = \left\{ \bar{a} \in {A_{\mathbf{t}}}^{arity(C)} : \mathfrak{A} \models \psi_C(\bar{a}) \right\}$. Due to the convention that structures do not have an empty universe, $\mathbf{t}^\star(\mathfrak{A})$ is defined iff $\mathfrak{A} \models \exists x \phi(x)$.

▶ **Example 5.1** (k-bounded structures)**.** Let $\mathcal{A}$ be a vocabulary. Recall from Section 4.2.2 that k-bounded $\mathcal{A}$-structures $\mathfrak{A}$ can be encoded in terms of $\mathcal{R}$-functional structures enriched with unary predicates encoding 2-types. For simplicity we assume $\mathcal{A}$ consists of binary relation symbols only. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ and let $\mathcal{X} = \{P_\lambda \mid \lambda \in \mathcal{R}\text{-MsgTypes}(\mathcal{A})\}$. Let $\mathbf{t}_{k-b.s.} = \langle \phi, \psi_C : C \in \mathcal{A} \rangle$ be the translation scheme for $\mathcal{A} \cup \mathcal{R}$ over $\mathcal{X} \cup \mathcal{R}$ given as follows: $\phi(x_0) = true$; for every $R_j \in \mathcal{R}$, $\psi_{R_j}(x_0, x_1) = R_j(x_0, x_1)$; and for every $C \in \mathcal{A}$, $\psi_C(x_0, x_1) = \bigvee_{i=1}^{k} (dir_{1,i} \vee dir_{2,i})$, where $dir_{j,i} = \bigvee_{\lambda \in \mathrm{Range}_j} R_i(x_0, x_1) \wedge P_\lambda(x_{j-1})$ for $j \in \{1, 2\}$. The inner disjunction $\bigvee_{\lambda \in \mathrm{Range}_1}$ in $dir_{1,i}$ ranges over message types $\lambda \in \mathcal{R}$-MsgTypes($\mathcal{A}$) such that both $C(x, y)$ and $R_i(x, y)$ belong to λ. The disjunction over Range_2 in $dir_{2,i}$ is similar except that here we require $C(y, x)$ and $R_i(x, y)$ to belong to λ.

The induced transduction $\mathbf{t}^\star_{k-b.s.}$ is the operation which takes encodings given as $(\mathcal{X} \cup \mathcal{R})$-structures to their corresponding k-bounded $(\mathcal{A} \cup \mathcal{R})$-structures. Given a $(\mathcal{X} \cup \mathcal{R})$-structure $\mathfrak{B}$, $\mathbf{t}^\star_{k-b.s.}(\mathfrak{B})$ has the same universe as $\mathfrak{B}$ since $\phi = true$. The reducts of $\mathbf{t}^\star_{k-b.s.}(\mathfrak{B})$ and $\mathfrak{B}$ to $\mathcal{R}$ are equal. There is a C-edge from an element a to an element b in $\mathbf{t}^\star_{k-b.s.}(\mathfrak{B})$ if there is $e \in \{(a, b), (b, a)\}$ whose 2-type is a $\mathcal{R}$-message type and the source of e is annotated with a predicate P_λ such that λ contains C in the relevant direction (i.e., $C(x, y) \in \lambda$ if the source of e is a, and $C(y, x) \in \lambda$ otherwise).

▶ **Lemma 5.2** (Fundamental property of translation schemes)**.** *Let $\mathbf{t}$ be a translation scheme for $\mathcal{C}_2$ over $\mathcal{C}_1$. There is a computable function $\mathbf{t}^\sharp$ from* MSO($\mathcal{C}_2$)*-sentences to* MSO($\mathcal{C}_1$)*-sentences such that for every $\mathcal{C}_1$-structure $\mathfrak{A}$ for which $\mathbf{t}^\star(\mathfrak{A})$ is defined and for every* MSO($\mathcal{C}_2$)*-sentence θ, $\mathfrak{A} \models \mathbf{t}^\sharp(\theta)$ if and only if $\mathbf{t}^\star(\mathfrak{A}) \models \theta$. We call $\mathbf{t}^\sharp$ the* **induced translation**.

For an MSO($\mathcal{C}_2$)-sentence ζ, $\mathbf{t}^\sharp$ substitutes the relation symbols $C \in \mathcal{C}_2$ in ζ with the formulas ψ_C, requires that each of the free variables satisfies ϕ, and relativizes the quantification to ϕ. An inductive definition of $\mathbf{t}^\sharp$ is given in Definition 3.2 of [20]. It is not difficult to extend the inductive definition of $\mathbf{t}^\sharp$ with the counting quantifiers (See Appendix C).

▶ **Example 5.3.** Let $\mathcal{E}$ be a vocabulary consisting of a single binary relation symbol E. The translation scheme $\mathbf{t}_{sym} = \langle \phi, \psi_E \rangle$ for $\mathcal{E}$ over $\mathcal{E}$ is given by $\phi(x) = E(x, x)$ and $\phi_E(x, y) = E(x, y) \vee E(y, x)$. The induced transduction $\mathbf{t}^\star_{sym}$ maps an $\mathcal{E}$-structure $\mathfrak{E}$ to the symmetric closure of the substructure of $\mathfrak{E}$ consisting of elements with self-loops. For example, for the structure $\mathfrak{E}_n$ with universe $[n]$ in which E is interpreted as the natural linear order $\leq$ of $[n]$, $\mathbf{t}^\star_{sym}(\mathfrak{E}_n)$ is $\langle [n], [n] \times [n] \rangle$. Let $full = \forall x \forall y\, E(x, y)$. By Lemma 5.2, $\mathbf{t}^\star_{sym}(\mathfrak{E}_n) \models full$ iff $\mathfrak{E}_n \models \mathbf{t}^\sharp_{sym}(full) = \forall x\ (E(x, x) \rightarrow (\forall y\, (E(y, y) \rightarrow (E(x, y) \vee E(y, x)))))$.

[5] All translation schemes in this paper are scalar (i.e. non-vectorized). In the notation of [6], a translation scheme is a parameterless non-copying MSO-definition scheme with precondition formula $(x \approx x)$.

▶ **Lemma 5.4** (Quantifier-free translation schemes and C^2)**.** *Let* $\mathbf{t}$ *be a quantifier-free translation scheme for* $\mathcal{C}_2$ *over* $\mathcal{C}_1$*. The induced translation* $\mathbf{t}^\sharp$ *maps* $C^2(\mathcal{C}_2)$*-formulas to* $C^2(\mathcal{C}_1)$*-formulas.*

For the proof of Lemma 3.3 we need a translation scheme for $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structures $\mathfrak{M}$ whose reducts $\mathfrak{M}|_{\mathcal{C}_{bnd}}$ have tree-width at most k. In order that α and β be mapped to an $\mathrm{MSO}(\mathcal{D}_{bnd})$-sentence and a $C^2(\mathcal{C}_{unb})$-sentence respectively, we need that the translation scheme satisfy some additional properties.

▶ **Lemma 5.5.** *Let* $\mathcal{C}_{bnd}$ *and* $\mathcal{C}_{unb}$ *be vocabularies such that* $\mathcal{C}_{bnd} \cap \mathcal{C}_{unb}$ *only contains unary relation symbols. There exist the following effectively computable objects: (1) a vocabulary* $\mathcal{D}_{bnd}$ *consisting of the binary relation symbol* s *and unary relation symbols only, (2) a translation scheme* $\mathbf{tr} = \langle \phi, \psi_C : C \in \mathcal{C}_{bnd} \cup \mathcal{C}_{unb} \rangle$ *for* $\mathcal{C}_{bnd} \cup \mathcal{C}_{unb}$ *over* $\mathcal{D}_{bnd} \cup \mathcal{C}_{unb}$*, and (3) an* $\mathrm{MSO}(\mathcal{D}_{bnd})$*-sentence dom, such that:*

- **(a)** *ϕ is quantifier-free over $\mathcal{D}_{bnd}$,*
- **(b)** *For every relation symbol $C \in \mathcal{C}_{unb}$, ψ_C is quantifier-free.*
- **(c)** *For every relation symbol $C \in \mathcal{C}_{bnd}$, ψ_C is an $\mathrm{MSO}(\mathcal{D}_{bnd})$-formula.*
- **(d)** *Let $\mathcal{K}$ be the class of $(\mathcal{D}_{bnd} \cup \mathcal{C}_{unb})$-structures in which s is interpreted as a binary tree and which satisfy dom. The image of $\mathcal{K}$ under $\mathbf{tr}^\star$ is exactly the class of $(\mathcal{C}_{bnd} \cup \mathcal{C}_{unb})$-structures $\mathfrak{M}$ such that $\mathfrak{M}|_{\mathcal{C}_{bnd}}$ has tree-width at most k.*

We use the translation scheme from Example 5.1 to encode k-bounded structures using $\mathcal{R}$-functional structures. (Note that structures of tree-width k are k-bounded.) We encode $\mathcal{R}$-functional structures as annotated trees using the inductive construction of k-trees in Section 2. The proof of Lemma 5.5 bears technical similarities to the discussion in [7] of graphs of bounded tree-width.

We are now ready to prove Lemma 3.3. By Lemma 5.5(d) and Lemma 5.2, statement (i) in Lemma 3.3 holds iff there is a $(\mathcal{D}_{bnd} \cup \mathcal{C}_{unb})$-structure $\mathfrak{N}$ such that $s^{\mathfrak{N}}$ is a binary tree and $\mathfrak{N} \models dom \wedge \mathbf{tr}^\sharp(\alpha) \wedge \mathbf{tr}^\sharp(\beta)$. Let $\alpha' = dom \wedge \mathbf{tr}^\sharp(\alpha)$ and $\beta' = \mathbf{tr}^\sharp(\beta)$. By Lemma 5.5(c) and the definition of $\mathbf{tr}^\sharp$, $\alpha' \in \mathrm{MSO}(\mathcal{D}_{bnd})$. Let $\mathbf{tr}|_{\mathcal{C}_{unb}}$ be the translation scheme for $\mathcal{C}_{unb}$ over $\mathcal{D}_{bnd} \cup \mathcal{C}_{unb}$ which agrees with $\mathbf{tr}$ on all formulas, i.e. $\mathbf{tr}|_{\mathcal{C}_{unb}} = \langle \phi, \psi_C : C \in \mathcal{C}_{unb} \rangle$. The image of $\mathbf{tr}|^\star_{\mathcal{C}_{unb}}$ on a structure $\mathfrak{M}$ is the $\mathcal{C}_{unb}$-reduct of the image of $\mathbf{tr}^\star$ on $\mathfrak{M}$. Since $\beta \in C^2(\mathcal{C}_{unb})$, $\mathbf{tr}|_{\mathcal{C}_{unb}}{}^\sharp(\beta)$ is well-defined and $\mathbf{tr}|_{\mathcal{C}_{unb}}{}^\sharp(\beta) = \beta'$. By Lemma 5.5(a,b), $\mathbf{tr}|_{\mathcal{C}_{unb}}$ is a quantifier-free translation scheme, implying that $\beta' \in C^2(\mathcal{D}_{bnd} \cup \mathcal{C}_{unb})$ by Lemma 5.4.

6 From MSO to C^2 on Binary Trees

The purpose of this section is to show that, on structures consisting only of a binary tree and additional unary relations, every MSO-sentence can be rewritten to a C^2-sentence which is equi-satisfiable and whose length is linear in the length of the input MSO-sentence. We start by introducing some tools from the literature.

▶ **Theorem 6.1** (Hintikka sentences)**.** *Let* $\mathcal{C}$ *be a vocabulary. For every* $q \in \mathbb{N}$ *there is a finite set* $\mathcal{HIN}_{\mathcal{C},q}$ *of* $\mathrm{MSO}(\mathcal{C})$ *sentences of quantifier rank* q *such that:*

1. *every $\epsilon \in \mathcal{HIN}_{\mathcal{C},q}$ has a model;*
2. *the conjunction of any two distinct sentences $\epsilon_1, \epsilon_2 \in \mathcal{HIN}_{\mathcal{C},q}$ is not satisfiable;*
3. *every $\mathrm{MSO}(\mathcal{C})$-sentence α of quantifier rank at most q is equivalent to exactly one finite disjunction of sentences $\mathcal{HIN}_{\mathcal{C},q}$;*
4. *every $\mathcal{C}$-structure $\mathfrak{A}$ satisfies exactly one sentence $\mathrm{hin}_{\mathcal{C},q}(\mathfrak{A})$ of $\mathcal{HIN}_{\mathcal{C},q}$.*

We may omit $\mathcal{C}$ or q from the subscript when they are clear from the context.

For a class of $\mathcal{C}$-structures $\mathcal{K}$ an n-ary operation Op over $\mathcal{C}$-structures is called *smooth over* $\mathcal{K}$, if for all $\mathfrak{A}_1, \ldots, \mathfrak{A}_n \in \mathcal{K}$, $\mathrm{hin}_{\mathcal{C},q}(Op(\mathfrak{A}_1, \ldots, \mathfrak{A}_n))$ depends only on $\mathrm{hin}_{\mathcal{C},q}(\mathfrak{A}_i)$: $i \in [n]$ and this dependence is computable[6]. We omit *"over $\mathcal{K}$"* when $\mathcal{K}$ consists of all $\mathcal{C}$-structures.

▶ **Theorem 6.2** (Smoothness)**.**

1. *The disjoint union is smooth.*
2. *For every quantifier-free translation scheme* $\mathbf{t}$*, the operation* $\mathbf{t}^\star$ *is smooth.*
3. *Let* $\mathfrak{T}_1 \overset{\rightarrow}{\circ} \mathfrak{T}_2$ *denote the tree obtained by adding* $\mathfrak{T}_2$ *as the child of* $\mathfrak{T}_1$*. That is,* $\mathfrak{T}_1 \overset{\rightarrow}{\circ} \mathfrak{T}_2$ *is obtained from the disjoint union of* $\mathfrak{T}_1$ *and* $\mathfrak{T}_2$ *by adding an edge from the root of tree* $\mathfrak{T}_1$ *to the root of tree* $\mathfrak{T}_2$*. The operation* $\overset{\rightarrow}{\circ}$ *is smooth on labeled trees.*[7]

For an in-depth introduction to Hintikka sentences and smoothness and references to proofs see [14, Chapter 3, Theorem 3.3.2] and [20] respectively.

We are now ready for the main lemma of this section.

Let $\mathrm{rt}(x)$ be the sentence $\forall y \neg s(y, x)$. This sentence defines the root of the binary tree s.

▶ **Lemma 6.3.** *Let* $q \in \mathbb{N}$ *and let* $\mathcal{C}$ *be a vocabulary which consists only of the binary relation symbol* s *and (possibly) additional unary relation symbols. For every* $\epsilon \in \mathcal{HIN}_{\mathcal{C},q}$*, let* C_ϵ *be a new unary relation symbol. Let* $\mathrm{hin}(\mathcal{C}, q)$ *be the vocabulary which extends* $\mathcal{C}$ *with* $\{C_\epsilon : \epsilon \in \mathcal{HIN}_{\mathcal{C},q}\}$*. There is a computable* $C^2(\mathrm{hin}(\mathcal{C}, q))$*-sentence* $\Theta^{\mathrm{hin}}_{\mathcal{C},q}$*, and for every* $\omega \in \mathrm{MSO}(\mathcal{C})$ *with* $\mathrm{qr}(\omega) = q$*, there exists a computable* C^2*-sentence* ω_{hin} *such that:*

(i) *Every* $\mathcal{C}$*-structure* $\mathfrak{T}_0$ *in which* $s^{\mathfrak{T}_0}$ *is a binary tree has a unique expansion* $\mathfrak{T}_1$ *to the vocabulary* $\mathrm{hin}(\mathcal{C}, q)$ *satisfying* $\mathfrak{T}_1 \models \Theta^{\mathrm{hin}}_{\mathcal{C},q}$*.*

(ii) *Moreover, for* $\mathfrak{T}_0$ *and* $\mathfrak{T}_1$ *as in (i),* $\mathfrak{T}_0 \models \omega$ *iff* $\mathfrak{T}_1 \models \omega_{\mathrm{hin}}$*.*

The C^2*-sentence* ω_{hin} *is*

$$\forall x \left(\mathrm{rt}(x) \to \bigvee_{\epsilon \in \mathcal{HIN}_{\mathcal{C},q} : \epsilon \models \omega} C_\epsilon(x) \right)$$

The sentence $\Theta^{\mathrm{hin}}_{\mathcal{C},q}$ is defined so that for every $\mathfrak{T}_0$ there is a unique expansion $\mathfrak{T}_1$ such that $\mathfrak{T}_1 \models \Theta^{\mathrm{hin}}_{\mathcal{C},q}$. For every u in the universe T_1 of $\mathfrak{T}_1$, we will have $u \in C_\epsilon^{\mathfrak{T}_1}$ iff the subtree $\mathfrak{T}_u$ of $\mathfrak{T}_0$ whose root is u satisfies $\mathfrak{T}_u \models \epsilon$. Using the smoothness of $\overset{\rightarrow}{\circ}$, whether an element of $\mathfrak{T}_1$ belongs to $C_\epsilon^{\mathfrak{T}_1}$ depends only on its children. This can be axiomatized in C^2. Lemma 3.4 follows from Lemma 6.3 with $q = \mathrm{qr}(\alpha)$, $\mathcal{D} = \mathrm{hin}(\mathcal{C}, q)$, and $\omega_{\mathrm{hin}} = \gamma$.

Appendix B spells out the proof of Lemma 6.3.

7 MSO with Cardinality Constraints

$\mathrm{MSO}^{\mathrm{card}}$ denotes the extension of MSO with atomic formulas called *cardinality constraints* $\sum_{i=1}^{r} |X_i| < \sum_{i=1}^{t} |Y_i|$, where the X_i and Y_i are MSO variables, and $|X|$ denotes the cardinality of X. Let WS1S ($\mathrm{WS1S}^{\mathrm{card}}$) be the weak monadic second order theory (with cardinality constraints) of the structure $\langle \mathbb{N}, +1, < \rangle$. Let $\mathrm{MSO}^{\exists \mathrm{card}} \subseteq \mathrm{MSO}^{\mathrm{card}}$ be the set of sentences ρ such that (1) ρ is of the form $\rho = \exists X_1 \cdots \exists X_m \omega$, and (2) only the $X_1, \ldots, X_m$ participate in cardinality constraints.

▶ **Theorem 7.1.** *Given a sentence* $\rho \in \mathrm{MSO}^{\exists \mathrm{card}}$*, it is decidable*

[6] Smooth operations here are called *effectively smooth* in [20].

[7] The smoothness of $\overset{\rightarrow}{\circ}$ can be shown using the smoothness of the *fusion* operation, see the discussion in Sections 3 and 4 of [20].

(A) *whether* $\langle \mathbb{N}, +1, < \rangle \models \rho$, *and*
(B) *whether* ρ *is satisfiable by a finite structure of bounded tree-width.*

We present the proof idea here, and the proof of Theorem 7.1(B) below. The proof of Theorem 7.1(A) is in Appendix A.

The proof of Theorem 7.1 follows from Theorem 3.1. The main observation needed for (B) is that cardinality constraints can be expressed in terms of injective functions, which are axiomatizable in C^2. (A) is reducible to (B). The main observations for (A) are:

1. that $X_1, \ldots, X_m$ are contained in the substructure $\mathfrak{A}_1$ of $\langle \mathbb{N}, +1, < \rangle$ generated by $\{0, \ldots, \ell\}$ for some $\ell \in \mathbb{N}$,
2. that substructure $\mathfrak{A}_2$ of $\langle \mathbb{N}, +1, < \rangle$ generated by $\mathbb{N} - \{0, \ldots, \ell\}$ is isomorphic to $\langle \mathbb{N}, +1, < \rangle$, and therefore $\mathfrak{A}_2$ and $\langle \mathbb{N}, +1, < \rangle$ have the same weak monadic second order theory,
3. that the weak monadic second order theory of $\langle \mathbb{N}, +1, < \rangle$ is decidable,
4. and that $\langle \mathbb{N}, +1, < \rangle$ is a transduction $\mathbf{t}$ of $\mathfrak{A}_1 \sqcup \mathfrak{A}_2$.

It remains to use that the disjoint union $\sqcup$ is smooth.

Proof of Theorem 7.1(B). Let ρ be a $\mathrm{MSO}^{\exists \mathrm{card}}$ sentence, i.e. the outermost block of quantifiers in ρ is existential and only variables from the outermost blocks may appear in cardinality constraints. For simplicity we consider $\rho = \exists X_1 \exists X_2 \omega$ with only two quantifiers in the outermost block. W.l.o.g. the only cardinality constraint in ω is $|X_1| < |X_2|$. By a slight abuse of notation, we sometimes treat X_1 and X_2 as unary relation symbols. Let $\mathcal{C}_{bnd}$ extend the vocabulary of ρ with new unary relation symbols X_1,X_2,W_{img},W_{dom}. Let $\mathcal{C}_{unb}$ extend $\mathcal{C}_{bnd}$ with a new binary relation symbol B. Finite satisfiability of ω by a structure $\mathfrak{M}$ such that $\mathrm{tw}(\mathfrak{M}) \leq k$ can be reduced to finite satisfiability of a sentence $\alpha \wedge \beta$, $\alpha \in \mathrm{MSO}(\mathcal{C}_{bnd})$ and $\beta \in C^2(\mathcal{C}_{unb})$, by a structure $\mathfrak{A}_1$ such that $\mathrm{tw}(\mathfrak{A}_1|_{\mathcal{C}_{bnd}}) \leq k$. Let β be the C^2-sentence $\beta = (\mathrm{inj}_{12} \vee \mathrm{inj}_{21}) \wedge \mathrm{dom} \wedge \mathrm{img}$, where:

- inj_{12} expresses that B is an injective function from X_1 to X_2,
- inj_{21} expresses that B is an injective function from X_2 to X_1,
- dom expresses that the domain of B is W_{dom},
- and img expresses that the image of B is W_{img}.

For every $\mathcal{C}_{unb}$-structure $\mathfrak{A}_1$, $|X_1^{\mathfrak{A}_1}| < |X_2^{\mathfrak{A}_1}|$ iff $W_{\mathrm{dom}}^{\mathfrak{A}_1} = X_1^{\mathfrak{A}_1}$ and $X_2^{\mathfrak{A}_1} \setminus W_{\mathrm{img}}^{\mathfrak{A}_1} \neq \emptyset$. Let α be obtained from ω by substituting every $|X_1| < |X_2|$ by

$$\forall x \, (X_1(x) \leftrightarrow W_{\mathrm{dom}}(x)) \wedge \exists x \, (\neg W_{\mathrm{img}}(x) \wedge X_2(x))$$

For any $\mathcal{C}_{bnd}$-structure $\mathfrak{A}_0$ with $\mathrm{tw}(\mathfrak{A}_0) \leq k$, $\mathfrak{A}_0 \models \omega$ iff there is an expansion $\mathfrak{A}_1$ of $\mathfrak{A}_0$ such that $\mathfrak{A}_1 \models \alpha \wedge \beta$. The treatment of other cardinality constraints $\sum_{i=1}^{r} |X_i| < \sum_{i=1}^{t} |Y_i|$ is similar; it is helpful to assume w.l.o.g. that the sets X_i and Y_i are pairwise disjoint. ◀

▶ Remark. While we assumed for simplicity in (B) that X_1 and X_2 range over subsets of the universe, it is not hard to extend the proof to the case that X_1 and X_2 are guarded second order variables which range over subsets of any relation in the structure. This is true since we can use the translation scheme $\mathbf{tr}$ from Lemma 5.5 to obtain the structures of tree-width at most k as the image of $\mathbf{tr}^\star$ of labeled trees; X_1 and X_2 then translate naturally to monadic second order variables.

Acknowledgements. We are grateful to the referees for their detailed comments on the presentation of the paper.

References

1 S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

2 F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic handbook*. Cambridge University Press, 2003.

3 D. Calvanese, T. Kotek, M. Šimkus, H. Veith, and F. Zuleger. Shape and content. In *Integrated Formal Methods*, pages 3–17. Springer, 2014.

4 W. Charatonik and P. Witkowski. Two-variable logic with counting and trees. In *LICS*, pages 73–82. IEEE, 2013.

5 B. Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.

6 B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

7 B. Courcelle and J. A. Makowsky. Fusion in relational structures and the verification of monadic second-order properties. *Mathematical Structures in Computer Science*, 12(02):203–235, 2002.

8 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006.

9 E. Grädel. On the restraining power of guards. *The Journal of Symbolic Logic*, 64(04):1719–1742, 1999.

10 E. Grädel and M. Otto. On logics with two variables. *Theoretical computer science*, 224(1):73–113, 1999.

11 E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *LICS*, pages 306–317. IEEE, 1997.

12 Erich Grädel, Martin Otto, and Eric Rosen. Undecidability results on two-variable logics. In *STACS 97, 14th Annual Symposium on Theoretical Aspects of Computer Science, Lübeck, Germany, February 27 – March 1, 1997, Proceedings*, pages 249–260, 1997.

13 J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm. *Mona: Monadic second-order logic in practice*. Springer, 1995.

14 W. Hodges. Model theory. In *Encyclopedia of Mathematics and its Applications*, volume 42. Cambridge University Press, 1993.

15 F. Klaedtke and H. Rueß. Monadic second-order logics with cardinalities. In *Automata, Languages and Programming*, pages 681–696. Springer, 2003.

16 T. Kotek and J. A. Makowsky. Connection matrices and the definability of graph parameters. *Logical Methods in Computer Science*, 10(4), 2014.

17 T. Kotek, M. Simkus, H. Veith, and F. Zuleger. Extending ALCQIO with trees. In *Logic in Computer Science LICS*, pages 511–522, 2015.

18 L. Lovász. *Large networks and graph limits*, volume 60. AMS, 2012.

19 P. Madhusudan, P. S. Thiagarajan, and Shaofa Yang. The MSO theory of connectedly communicating processes. In *FSTTCS*, pages 201–212, 2005.

20 J.A. Makowsky. Algorithmic uses of the Feferman–Vaught theorem. *Annals of Pure and Applied Logic*, 126(1):159–213, 2004.

21 G. Nelson and D.C. Oppen. Simplification by cooperating decision procedures. *TOPLAS*, 1(2):245–257, 1979.

22 A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

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 A.J.A. Robinson and A. Voronkov. *Handbook of automated reasoning*. Elsevier, 2001.

25 D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic*, 53(2):169–195, 1991.

26 W. Thomas. *Languages, automata, and logic.* Springer, 1997.

27 M.Y. Vardi. Why is modal logic so robustly decidable? *Descriptive complexity and finite models*, 31:149–184, 1996.

A Appendix: Proof of Theorem 7.1(A)

The unary function $+1$ is the *successor relation* of $\mathbb{N}$ and interprets the binary relation symbol suc. The binary relation $<$ is the *natural order relation* on $\mathbb{N}$. In the proof of (A) we will use the theory of Hintikka sentences as presented in Section 6 with one caveat, namely that instead of restricting to finite structures, we allow arbitrary structures. Theorems 6.1 and 6.2 hold for arbitrary structures. Theorems 6.1 guarantees the existence of a set $\mathcal{HIN}^{\text{arb}}_{\mathcal{C},q}$ analogously to $\mathcal{HIN}_{\mathcal{C},q}$ for arbitrary structures. For every $\mathcal{C}$-structure $\mathfrak{A}$, Theorem 6.1 guarantees the existence of a sentence $\text{hin}^{\text{arb}}_{\mathcal{C},q}(\mathfrak{A})$ of $\mathcal{HIN}^{\text{arb}}_{\mathcal{C},q}$ analogously to $\text{hin}_{\mathcal{C},q}(\mathfrak{A})$. For the rest of the section, we omit the superscript arb to simplify notation.

Consider $\rho = \exists X_1 \exists X_2 \omega$ for the vocabulary $\mathcal{C}_{\mathbb{N}}$ of $\langle \mathbb{N}, +1, < \rangle$. Let $\alpha \in \text{MSO}(\mathcal{C}_{bnd})$ and $\beta \in C^2(\mathcal{C}_{unb})$ be as discussed in the proof of (B) above. The following are equivalent:

1. $\langle \mathbb{N}, +1, < \rangle \models \rho$
2. There are *finite* unary relations U_1 and U_2 such that $\langle \mathbb{N}, +1, <, U_1, U_2 \rangle \models \omega$. U_1 and U_2 interpret X_1 and X_2 respectively.
3. There are *finite* unary relations U_1 and U_2 and an expansion $\mathfrak{A}$ of $\langle \mathbb{N}, +1, <, U_1, U_2 \rangle$ such that $\mathfrak{A} \models \alpha \wedge \beta$. $\mathfrak{A}$ expands $\langle \mathbb{N}, +1, <, U_1, U_2 \rangle$ with interpretations of the symbols in $\mathcal{D}_{unb} = \{B, W_{\text{dom}}, W_{\text{img}}\}$.
4. $\rho' = \exists X_1 \exists X_2 (\alpha \wedge \beta)$ is satisfiable by an expansion of $\langle \mathbb{N}, +1, < \rangle$ with interpretations for the symbols of $\mathcal{D}_{unb}$.

We have *1. iff 2.* and *3. iff 4.* by the semantics of $\exists X_1 \exists X_2$ in weak monadic second order logic. We have *2. iff 3.* similarly to the discussion of α and β in the proof of (A) above. The rest of the proof is devoted to proving that 4. is decidable.

Observe that by the definition of β, and in weak monadic second order X_1 and X_2 are quantified to be finite sets, B is axiomatized to be a function with finite domain, and W_{dom} and W_{img} are finite. Hence models $\mathfrak{A}$ of ρ' can be decomposed into a finite part containing $B^{\mathfrak{A}}$, and an infinite part isomorphic to an expansion of $\langle \mathbb{N}, +1, < \rangle$ in which the symbols of $\mathcal{D}_{unb}$ as are interpreted as empty sets. We will use a similar decomposition, but first we want to move from the structure $\langle \mathbb{N}, +1, < \rangle$ and its expansions to $\langle \mathbb{N}, +1 \rangle$ and its expansions. There is a translation scheme $\mathbf{t}_<$ such that for every structure $\mathfrak{A} = \langle \mathbb{N}, +1, B^{\mathfrak{A}}, W^{\mathfrak{A}}_{\text{dom}}, W^{\mathfrak{A}}_{\text{img}} \rangle$, $\mathbf{t}^\star_<(\langle \mathfrak{A} \rangle) = \langle \mathfrak{A}, < \rangle$, where $\langle \mathfrak{A}, < \rangle$ is the expansion of $\mathfrak{A}$ with $<$. This is true since $<$ is MSO definable from $+1$. We have that ρ' is satisfied by an expansion of $\langle \mathbb{N}, +1, < \rangle$ iff $\mathbf{t}^\sharp_<(\rho')$ is satisfied by the same expansion of $\langle \mathbb{N}, +1 \rangle$.

Now we turn to the decomposition of models of $\mathbf{t}^\sharp_<(\rho')$ into a finite part containing $B^{\mathfrak{A}}$, and an infinite part isomorphic to a $\mathcal{D}_{unb}$-expansion of $\langle \mathbb{N}, +1 \rangle$. Let $\mathcal{D}_{unb+} = \langle \mathcal{D}_{unb}, + \rangle$ and note that $\mathcal{D}_{unb+}$ is the vocabulary of $\mathbf{t}^\sharp_<(\rho')$. For every $\mathcal{D}_{unb+}$-expansion $\mathfrak{P}$ of $\langle \mathbb{N}, +1 \rangle$ and every $n \in \mathbb{N}$, let $\mathfrak{P}_{1,n}$ and $\mathfrak{P}_{n,\infty}$ be the substructures of $\mathfrak{P}$ generated by $[n]$ and $\mathbb{N} \backslash [n-1]$ respectively. There is a translation scheme $\mathbf{u}$ such that $\mathbf{u}^\star(\mathfrak{P}_{1,n} \sqcup \mathfrak{P}_{n+1,\infty}) = \mathfrak{P}$ if $B^{\mathfrak{P}} \subseteq [n] \times [n]$. $\mathbf{u}$ existentially quantifies the set $[n]$ (which is the only non-empty finite set closed under suc and its inverse), 1 and n (as the first and last elements of $[n]$) and $n+1$ (as the only element without a suc-predecessor except for 1) and adds the edge $(n, n+1)$ to suc. We have $\mathfrak{P} \models \mathbf{t}^\sharp_<(\rho')$ iff $\mathfrak{P}_{1,n} \sqcup \mathfrak{P}_{n+1,\infty} \models \mathbf{u}^\sharp(\mathbf{t}^\sharp_<(\rho'))$. Note that the vocabulary of $\mathbf{u}^\sharp(\mathbf{t}^\sharp_<(\rho'))$ is $\mathcal{D}_{unb+}$. Let q be the quantifier rank of $\mathbf{u}^\sharp(\mathbf{t}^\sharp_<(\rho'))$.

The Hintikka sentence $\mathrm{hin}(\mathfrak{P}_{n+1,\infty})$ of quantifier rank q of $\mathfrak{P}_{n+1,\infty}$ is uniquely defined since $\mathfrak{P}_{n+1,\infty}$ is isomorphic to the expansion of $\langle \mathbb{N}, +1\rangle$ with empty sets. Moreover, $\mathrm{hin}(\mathfrak{P}_{n+1,\infty})$ is computable using that the theory of $\langle \mathbb{N}, +1\rangle$ is decidable. Hence, by the smoothness of the disjoint union, for every Hintikka sentence $\epsilon \in \mathcal{HIN}_{\mathcal{D}_{unb+},q}$ there is a computable set $E_\epsilon \subseteq \mathcal{HIN}_{\mathcal{D}_{unb+},q}$ such that $\mathrm{hin}(\mathfrak{P}_{1,n} \sqcup \mathfrak{P}_{n+1,\infty}) = \epsilon$ iff $\mathrm{hin}(P_{1,n}) \in E_\epsilon$. Then $\mathfrak{P}_{1,n} \sqcup \mathfrak{P}_{n+1,\infty} \models \mathbf{u}^\sharp(\mathbf{t}^\sharp_<(\rho'))$ iff $\mathfrak{P}_{1,n}$ satisfies the sentence $\bigvee_{(\epsilon,\epsilon')} \epsilon'$, where the $\bigvee_{(\epsilon,\epsilon')}$ ranges over pairs (ϵ, ϵ') such that (1) $\epsilon \in \mathcal{HIN}_{\mathcal{D}_{unb+},q}$, (2) $\epsilon \models \mathbf{u}^\sharp(\mathbf{t}^\sharp_<(\rho'))$, and (3) $\epsilon' \in E_\epsilon$. Hence, ρ' is satisfiable by an expansion of $\langle \mathbb{N}, +1, <\rangle$ iff $\bigvee_{(\epsilon,\epsilon')} \epsilon'$ is satisfiable by a finite structure in which suc is interpreted as a successor relation (i.e., as a simple directed path on the whole universe). Let suc-rel be the weak MSO-sentence such that the interpretation of suc is a successor relation. By Theorem 7.1(B), it is decidable whether, $\bigvee_{(\epsilon,\epsilon')} \epsilon' \wedge$ suc-rel is finitely satisfiable using that the class of simple directed paths annotated with unary relations has tree-width 1.

B Appendix: Proof of Lemma 6.3

For a leaf b, $\mathrm{hin}_q(\mathfrak{T}_b)$ depends only on the unary relations which b satisfies. By Theorem 6.2, for a vertex b with one child b_0 (two children b_0, b_1), $\mathrm{hin}_q(\mathfrak{T}_b)$ depends only on the unary relations which b satisfies and on $\mathrm{hin}_q(\mathfrak{T}_{b_0})$ (on $\mathrm{hin}_q(\mathfrak{T}_{b_0})$ and $\mathrm{hin}_q(\mathfrak{T}_{b_1})$).

Let

$$\Theta^{\mathrm{hin}}_{\mathcal{C},q} = \mathrm{part} \wedge \mathrm{leaves} \wedge \mathrm{ints}_1 \wedge \mathrm{ints}_2$$

where part says that $\{C_\epsilon : \epsilon \in \mathcal{HIN}_{\mathcal{C},q}\}$ partition the universe, and leaves, ints_1, and ints_2 define the C_ϵ for the leaves respectively the internal vertices of $\mathfrak{T}_0$ with one or two children.

We give part, leaves, ints_1 and ints_2 below. There are C^2 formulas $\mathrm{leaf}(x)$, $\mathrm{int}_1(x)$, and $\mathrm{int}_2(x)$, which express that x is a leaf, has one child, or has two children, respectively. Let

$$\mathrm{part} = \forall x \left(\left(\bigvee_{\epsilon \in \mathcal{HIN}_{\mathcal{C},q}} C_\epsilon(x) \right) \wedge \left(\bigwedge_{\epsilon_1 \neq \epsilon_2 \in \mathcal{HIN}_{\mathcal{C},q}} (\neg C_{\epsilon_1}(x) \vee \neg C_{\epsilon_2}(x)) \right) \right) .$$

For $\mathcal{U} \subseteq \mathrm{un}(\mathcal{C})$, let $\mathfrak{O}_\mathcal{U}$ be a $\mathcal{C}$-structure with universe O of size 1 satisfying that $U^{\mathfrak{O}_\mathcal{U}} = \emptyset$ iff $U \in \mathcal{U}$, and $\mathrm{root}^{\mathfrak{O}_\mathcal{U}} = O$. Let

$$\begin{aligned}
\mathrm{ints}_1 &= \forall x \left(\mathrm{int}_1(x) \to \bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2) \in \mathrm{Range}_3} ((\mathrm{this}_\mathcal{U}(x) \wedge \mathrm{child}_{\epsilon_2}(x)) \to C_{\epsilon_1}(x)) \right) \\
\mathrm{this}_\mathcal{U}(x) &= \bigwedge_{U \in \mathrm{un}(\mathcal{C}), U \notin \mathcal{U}} U(x) \wedge \bigwedge_{U \in \mathrm{un}(\mathcal{C}), U \in \mathcal{U}} \neg U(x) \\
\mathrm{child}_{\epsilon_2}(y) &= \exists y\, s(x,y) \wedge C_{\epsilon_2}(y)
\end{aligned}$$

The conjunction $\bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2) \in \mathrm{Range}_3}$ ranges over tuples $(\mathcal{U}, \epsilon_1, \epsilon_2)$ such that $\mathcal{U} \subseteq \mathrm{un}(\mathcal{C})$, $\epsilon_1, \epsilon_2 \in \mathcal{HIN}_{\mathcal{C},q}$, and for every structure $\mathfrak{A} \in \mathcal{K}_{\mathrm{root}}$ with $\mathrm{hin}(\mathfrak{A}) = \epsilon_2$, $\mathrm{hin}(\mathfrak{O}_\mathcal{U} \overrightarrow{\circ}\, \mathfrak{A}) = \epsilon_1$.

We define:

$$\mathrm{leaves} = \forall x \left(\mathrm{leaf}(x) \to \bigwedge_{(\mathcal{U},\epsilon_1) \in \mathrm{Range}_4} \mathrm{this}_\mathcal{U}(x) \to C_{\epsilon_1}(x) \right)$$

The conjunction $\bigwedge_{(\mathcal{U},\epsilon_1)\in \mathrm{Range}_4}$ ranges over tuples $(\mathcal{U},\epsilon_1)$ such that $\mathcal{U} \subseteq \mathrm{un}(\mathcal{C})$, $\epsilon_1 \in \mathcal{HIN}_{\mathcal{C},q}$, and $\mathrm{hin}(\mathfrak{O}_\mathcal{U}) = \epsilon_1$. Finally, ints_2 is defined as follows:

$$\begin{aligned}
\mathrm{ints}_2 &= \forall x(\mathrm{int}_2(x) \to (\mathrm{intdist}(x) \wedge \mathrm{intsame})) \\
\mathrm{intdist}(x) &= \bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2,\epsilon_3)\in \mathrm{Range}_5} ((\mathrm{this}_\mathcal{U}(x) \wedge \mathrm{child}_{\epsilon_2}(x) \wedge \mathrm{child}_{\epsilon_3}(x)) \to C_{\epsilon_1}(x)) \\
\mathrm{intsame}(x) &= \bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2)\in \mathrm{Range}_6} ((\mathrm{this}_\mathcal{U}(x) \wedge \mathrm{children}_{\epsilon_2}(x)) \to C_{\epsilon_1}(x)) \\
\mathrm{children}_{\epsilon_2}(x) &= \exists^{=2} y\,(s(x,y) \wedge C_{\epsilon_2}(y))
\end{aligned}$$

The conjunction $\bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2,\epsilon_3)\in \mathrm{Range}_5}$ in $\mathrm{intdist}(x)$ ranges over $\mathcal{U} \subseteq \mathrm{un}(\mathcal{C})$ and $\epsilon_1,\epsilon_2,\epsilon_3 \in \mathcal{HIN}_{\mathcal{C},q}$ such that $\epsilon_2 \neq \epsilon_3$, and for every two structure $\mathfrak{A},\mathfrak{B} \in \mathcal{K}_{\mathrm{root}}$ with $\mathrm{hin}(\mathfrak{A}) = \epsilon_2$ and $\mathrm{hin}(\mathfrak{B}) = \epsilon_3$, $\mathrm{hin}((\mathfrak{O}_\mathcal{U} \overset{\rightarrow}{\circ} \mathfrak{A}) \overset{\rightarrow}{\circ} \mathfrak{B}) = \epsilon_1$. The conjunction $\bigwedge_{(\mathcal{U},\epsilon_1,\epsilon_2)\in \mathrm{Range}_6}$ in $\mathrm{intsame}(x)$ ranges over $\mathcal{U} \subseteq \mathrm{un}(\mathcal{C})$ and $\epsilon_1,\epsilon_2 \in \mathcal{HIN}_{\mathcal{C},q}$ such that for every structure $\mathfrak{A} \in \mathcal{K}_{\mathrm{root}}$ with $\mathrm{hin}(\mathfrak{A}) = \epsilon_2$, $\mathrm{hin}((\mathfrak{O}_\mathcal{U} \overset{\rightarrow}{\circ} \mathfrak{A}) \overset{\rightarrow}{\circ} \mathfrak{A}') = \epsilon_1$, where $\mathfrak{A}'$ is isomorphic to $\mathfrak{A}$ over a disjoint universe.

By definition, $\Theta^{\mathrm{hin}}_{\mathcal{C},q} \in C^2$. Let $\mathfrak{T}_0$ be a $\mathcal{C}$-structure in which s is interpreted as a binary tree. Let $\mathfrak{T}_1$ be the expansion of $\mathfrak{T}_0$ such that, for every element b of the universe $T_1 = T_0$ of $\mathfrak{T}_1$, $b \in C^{T_1}_{\mathrm{hin}(\mathfrak{T}_b)}$, where $\mathfrak{T}_b$ is the subtree of $\mathfrak{T}_0$ rooted at b. In particular, for the root r of $\mathfrak{T}_0$, $r \in C^{T_1}_{\mathrm{hin}(\mathfrak{T}_r)} = C^{T_1}_{\mathrm{hin}(\mathfrak{T}_0)}$. The structure $\mathfrak{T}_1$ is the unique expansion of $\mathfrak{T}_0$ such that $\mathfrak{T}_1 \models \Theta^{\mathrm{hin}}_{\mathcal{C},q}$, hence (i) holds. Using the definition of ω_{hin}, (ii) holds.

Note that $\Theta^{\mathrm{hin}}_{\mathcal{C},q}$ is a sentence in the description logic $\mathcal{ALCQIO}$ (which is a sublogic of C^2) discussed in [17]. The same is true for the sentences ω_{hin}.

C Appendix: The induced translation $\mathbf{t}^\sharp$

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be vocabularies. Given a translation scheme $\mathbf{t} = \langle \phi, \psi_C : C \in \mathcal{C}_2\rangle$ for $\mathcal{C}_0$ over $\mathcal{C}_1$ we define the induced translation $\mathbf{t}^\sharp$ to be a function from $\mathrm{MSO}(\mathcal{C}_2)$-formulas to $\mathcal{C}_1$-formulas inductively as follows:

1. For $C \in \mathrm{un}(\mathcal{C}_2)$ or for monadic second order variables C, and for $\theta = C(x)$, we put
$$\mathbf{t}^\sharp(\theta) = \psi_C(x) \wedge \phi(x)$$
2. For $C \in \mathrm{bin}(\mathcal{C}_2)$ and $\theta = C(x,y)$, we put $\mathbf{t}^\sharp(\theta) = \psi_C(x,y) \wedge \phi(x) \wedge \phi(y)$
3. For $x \approx y$, we put $\mathbf{t}^\sharp(\theta) = x \approx y \wedge \phi(x) \wedge \phi(y)$
4. For the Boolean connectives the translation distributes, i.e.
   - if $\theta = \theta_1 \vee \theta_2$ then $t^\sharp(\theta) = (\mathbf{t}^\sharp(\theta_1) \vee \mathbf{t}^\sharp(\theta_2))$
   - if $\theta = \neg\theta_1$ then $t^\sharp(\theta) = \neg\mathbf{t}^\sharp(\theta_1)$
5. For the existential quantifiers, we relativize to ϕ:
   If $\theta = Q\,y\,\theta_1$, where $Q \in \{\exists\} \cup \{\exists^{\leq n}, \exists^{\geq n}, \exists^{=n} : n \in \mathbb{N}\}$, we put $\mathbf{t}^\sharp(\theta) = Q\,y\,(\phi(y) \wedge \mathbf{t}^\sharp(\theta_1))$
   If $\theta = \exists U\,\theta_1$, we put $\mathbf{t}^\sharp(\theta) = \exists U\,(\mathbf{t}^\sharp(\theta_1) \wedge \forall y\,U(y) \to \phi(y))$

We have somewhat simplified the presentation in [20, Definition 2.3] to fit our setting. On the other hand, we have extended the presentation in [20] by the counting quantifiers. This is important for Lemma 5.4 on C^2-translations schemes. (Note that for the Lemma 5.4 it is not enough to use that counting quantifiers are definable in MSO.)

Dependence Logic vs. Constraint Satisfaction

Lauri Hella[*1] **and Phokion G. Kolaitis**[†2]

1 **School of Information Sciences, University of Tampere, Finland**
`lauri.hella@uta.fi`
2 **University of California Santa Cruz and IBM Research – Almaden, USA**
`kolaitis@cs.ucsc.edu`

Abstract

During the past decade, dependence logic has emerged as a formalism suitable for expressing and analyzing notions of dependence and independence that arise in different scientific areas. The sentences of dependence logic have the same expressive power as those of existential second-order logic, hence dependence logic captures NP on the class of all finite structures. In this paper, we identify a natural fragment of universal dependence logic and show that, in a precise sense, it captures constraint satisfaction. This tight connection between dependence logic and constraint satisfaction contributes to the descriptive complexity of constraint satisfaction and elucidates the expressive power of universal dependence logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.1.3 Complexity Measures and Classes

Keywords and phrases Dependence logic, constraint satisfaction, computational complexity, expressive power

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.14

1 Introduction

Dependence logic is a formalism for expressing and analyzing notions of dependence and independence that are encountered across different areas of computer science and mathematics, from functional dependencies in relational databases to independence in linear algebra and in probability theory. Even though its origins can be traced back to Henkin quantifiers [10] and to independence-friendly logic [11], dependence logic was fully developed by Väänänen in his monograph [17], which became the catalyst for numerous subsequent investigations (see, e.g., [6, 7, 8, 13, 14]). The syntax of dependence logic uses *dependence atoms* as the main building blocks; these atoms assert that a functional dependency between variables holds, i.e., that a certain variable is a function of some other variables. The semantics of dependence logic uses sets of assignments, called *teams*, instead of single assignments of values to variables. In terms of expressive power and as regards sentences, dependence logic has the same expressive power as existential second-order logic [13]. Combined with Fagin's Theorem [4], this result implies that, on classes of finite structures, the sentences of dependence logic can express precisely all decision problems in NP.

Constraint satisfaction comprises a set of algorithmic problems that are ubiquitous in several different areas of computer science. An influential paper by Feder and Vardi [5] provided the impetus for an in-depth and still ongoing investigation of the connections

* The research of Lauri Hella was partially supported by a Professor Pool's Grant of the Finnish Cultural Foundation.
† The research of Phokion Kolaitis was partially supported by NSF Grant IIS-1217869.

© Lauri Hella and Phokion G. Kolaitis;
licensed under Creative Commons License CC-BY

between constraint satisfaction, computational complexity, logic, and universal algebra (see, e.g., [1, 9]). Feder and Vardi argued convincingly that, in its most general form, constraint satisfaction can be identified with the Homomorphism Problem: given two relational structures $\mathfrak{A}$ and $\mathfrak{B}$, is there a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$? Clearly, the Homomorphism Problem is NP-complete, since it contains, for example, 3-Satisfiability as a special case. Moreover, each fixed relational structure $\mathfrak{B}$ gives rise to the *non-uniform* constraint satisfaction problem CSP($\mathfrak{B}$): given a relational structure $\mathfrak{A}$, is there a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$? The computational complexity of each such problem depends on the structure $\mathfrak{B}$. Feder and Vardi conjectured that the family of all constraint satisfaction problems CSP($\mathfrak{B}$) exhibits the following dichotomy: for each $\mathfrak{B}$, either CSP($\mathfrak{B}$) is NP-complete or CSP($\mathfrak{B}$) is solvable in polynomial time. This conjecture remains open to date, in spite of concerted efforts by different groups of researchers that, so far, have established only special cases of it.

Feder and Vardi [5] also investigated the descriptive complexity of constraint satisfaction. To this effect, they identified a fragment of existential second-order logic, called *monadic monotone strict* NP *without inequality* or, in short, MMSNP, and showed that it captures, in a precise sense, the family CSP($\mathfrak{B}$) of all non-uniform constraint satisfaction problems. MMSNP consists of all sentences of existential second order logic that have the following properties (where it is assumed that all negation symbols occurring in the sentences have been pushed inward, so that they apply to atomic formulas only): (a) all second-order quantifiers are monadic; (b) all first-order quantifiers are universal; (c) no inequalities occur in the formula; (d) all occurrences of relation symbols from the underlying vocabulary are preceded by the negation symbol. MMSNP captures constraint satisfaction in the following way. First, it is easy to see that if $\mathfrak{B}$ is a relational structure, then CSP($\mathfrak{B}$) is expressible in MMSNP. Second, Feder and Vardi showed that every MMSNP-expressible problem is equivalent to a CSP($\mathfrak{B}$), for some relational structure $\mathfrak{B}$, under polynomial-time reductions (originally, this equivalence was proved under randomized polynomial-time reductions, which, however, were subsequently derandomized [15]). Note that Feder and Vardi also showed that if one of the aforementioned properties (a), (b), (c), (d) defining MMSNP is dropped, then every problem in NP is equivalent under polynomial-time reductions to a problem in the resulting fragment of existential second-order logic. Combined with Ladner's Theorem [16], this implies that if one of these four properties is dropped, then the resulting fragment can express decision problems that are neither NP-complete, nor solvable in polynomial time (unless P = NP).

As seen from the preceding discussion, dependence logic captures existential second-order logic, while constraint satisfaction is captured by a proper fragment of existential second-order logic. This state of affairs gives rise to the following question: is there a natural fragment of dependence logic that captures constraint satisfaction? In this paper, we show that this is indeed the case. In fact, we identify a fragment of a variant of dependence logic consisting of universal sentences and show that it can capture, in a precise sense, constraint satisfaction. In what follows in this section, we present a high-level description of our main results.

The building blocks of dependence logic, as developed by Väänänen, are dependence atoms $\mathrm{dep}(\boldsymbol{x}; y)$, where $\boldsymbol{x}$ is a tuple of variables and y is a single variable. A team (i.e., a set of assignments) satisfies such an atom if whenever two assignments in the team agree on the variables in $\boldsymbol{x}$, they must also agree on the variable y. Here, we introduce a variant of dependence atoms, which we call *uniform* dependence atoms; they are expressions of the form $\mathrm{udep}(x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n)$ with the following semantics: a team T satisfies $\mathrm{udep}(x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n)$ if there is a unary function f such that for every assignment s in T, we have that $s(\alpha_i) = f(s(x_i))$, for $1 \leq i \leq n$. Even though uniform dependence atoms have not been studied in their own right in earlier work on dependence logic, we believe that

they are very natural as they express scenarios in which n different observers use sensors or measuring instruments to collect data in different sites, and then each observer applies the same function to the data collected to obtain a value. As a concrete example, each x_i may represent a list of temperature values collected at site i at regular intervals of time each day, while α_i may stand for the maximum temperature at site i.

We consider *k-valued* uniform dependence atoms in which the variables $\alpha_1, \ldots, \alpha_n$ take values in a domain with k elements, for some fixed $k \geq 1$. We define the *universal monotone uniform dependence logic* $\forall$-MUD$[k]$ as the closure under universal quantification of all quantifier-free formulas that contain all k-valued uniform dependence atoms, all equalities between k-valued variables and constants, and all negated relational atoms, and are closed under disjunctions and conjunctions. The semantics of the logic $\forall$-MUD$[k]$ are given using teams as in (standard) dependence logic.

Our first main result asserts that every non-uniform constraint satisfaction problem CSP($\mathfrak{B}$) such that $\mathfrak{B}$ has a single relation is expressible by a sentence of $\forall$-MUD$[k]$, where k is the number of elements in the universe of $\mathfrak{B}$. Our second main result asserts that every sentence of $\forall$-MUD$[k]$, $k \geq 1$, is equivalent to a sentence of MMSNP. Since, as described earlier, every MMSNP-expressible problem is polynomial-time equivalent to some non-uniform constraint satisfaction problem [5] and since, as shown in [5] and in [15], every non-uniform constraint satisfaction problem is polynomial-time equivalent to some non-uniform constraint satisfaction problem on a structure with a single relation, our two main results imply that universal monotone uniform dependence logic captures, in a precise sense, all non-uniform constraint satisfaction problems CSP($\mathfrak{B}$).

Our results establish a tight connection between constraint satisfaction and a natural fragment of dependence logic. From the standpoint of constraint satisfaction, they contribute to the investigation of the descriptive complexity of constraint satisfaction. From the standpoint of dependence logic, they reveal that a dichotomy theorem for the computational complexity of the universal fragment of uniform dependence logic is as difficult as a dichotomy theorem for constraint satisfaction, which, to date, remains an elusive goal.

2 Background and Basic Notions

All structures considered in this paper are *finite* and *relational*. Thus, a vocabulary τ is a finite set of $\{R_1, \ldots, R_n\}$ of relation symbols, and the domain dom($\mathfrak{A}$) of each τ-structure $\mathfrak{A} = (\text{dom}(\mathfrak{A}), R_1^{\mathfrak{A}}, \ldots, R_n^{\mathfrak{A}})$ is assumed to be finite. However, to interpret k-valued dependence atoms, we add k constant symbols to the vocabulary; see Subsection 2.3 below. We will usually denote dom($\mathfrak{A}$) by A, dom($\mathfrak{B}$) by B, etc. For any integer $k \geq 1$, we will use the notation $[k] = \{1, \ldots, k\}$ throughout.

2.1 Constraint Satisfaction and MMSNP

A *homomorphism* between two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ is a function h from the universe A of $\mathfrak{A}$ to the universe B of $\mathfrak{B}$ such that for every relation symbol R of τ and every tuple $(a_1, \ldots, a_n)$ of elements of A, if $(a_1, \ldots, a_n) \in R^{\mathfrak{A}}$, then $(h(a_1), \ldots, h(a_n)) \in R^{\mathfrak{B}}$. Every τ-structure $\mathfrak{B}$ gives rise to the following constraint satisfaction problem CSP($\mathfrak{B}$):

Given a τ-structure $\mathfrak{A}$, is there a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$?

According to the usual practise, we identify the problem CSP($\mathfrak{B}$) with the class of its positive instances. Thus, we write $\mathfrak{A} \in \text{CSP}(\mathfrak{B})$, if the answer to the question above is "yes".

Clearly, each constraint satisfaction problem CSP($\mathfrak{B}$) is in NP. Moreover, numerous natural computational problems can be viewed as constraint satisfaction problems for a suitable choice of $\mathfrak{B}$. For example, if K_k is the *complete* graph on k nodes (i.e., K_k is the *k-clique*), $k \geq 2$, then CSP(K_k) is the k-Colorability problem. Furthermore, several variants of Satisfiability can be viewed as constraint satisfaction problems. We now give two such examples.

First, consider a vocabulary τ consisting of four ternary relation symbols R_0, R_1, R_2, R_3 and let $\mathfrak{B}$ be the τ-structure with universe $\{0,1\}$ and relations $R_0^{\mathfrak{B}} = \{0,1\}^3 \setminus \{(0,0,0)\}$, $R_1^{\mathfrak{B}} = \{0,1\}^3 \setminus \{(1,0,0)\}$, $R_2^{\mathfrak{B}} = \{0,1\}^3 \setminus \{(1,1,0)\}$, $R_3^{\mathfrak{B}} = \{0,1\}^3 \setminus \{(1,1,1)\}$. It is easy to see that CSP($\mathfrak{B}$) amounts to 3-Sat, where a 3CNF-formula φ is encoded as a τ-structure $\mathfrak{A}_\varphi$ with universe the set of its variables and where the relation $R_i^{\mathfrak{A}_\varphi}$ interpreting R_i consists of the triples of variables occurring in a clause with i negative literals, $i = 0, 1, 2, 3$.

Next, consider a vocabulary τ consisting of a single ternary relation symbol R and let $\mathfrak{B}$ be the τ-structure with universe $\{0,1\}$ and relation $R^{\mathfrak{B}} = \{(1,0,0), (0,1,0), (0,0,1)\}$. It is easy to see that CSP($\mathfrak{B}$) amounts to Positive 1-in-3 Sat: given a 3CNF-formula φ consisting entirely of positive clauses, is there a truth assignment t such that, for every clause c of φ, the assignment t makes true exactly one of the three variables of c? Here, φ is encoded as a τ-structure $\mathfrak{A}_\varphi$ with universe the set of its variables and where the relation $R^{\mathfrak{A}_\varphi}$ consists of all triples (x, y, z) of variables such that $(x \vee y \vee z)$ is a clause of φ.

As mentioned in the Introduction, Feder and Vardi [5] conjectured that, for every fixed τ-structure $\mathfrak{B}$, either CSP($\mathfrak{B}$) is NP-complete or CSP($\mathfrak{B}$) is solvable in polynomial time. Moreover, they showed that, for every τ-structure $\mathfrak{B}$, there is a structure $\mathfrak{B}'$ over a vocabulary consisting of a single binary relation such that CSP($\mathfrak{B}$) and CSP($\mathfrak{B}'$) are equivalent via polynomial-time reductions. Thus, to settle the Feder-Vardi conjecture, it is enough to settle it for structures with a single binary relation (i.e., for directed graphs).

Every constraint satisfaction problem CSP($\mathfrak{B}$) is expressible by a sentence of existential second-order logic that also obeys certain syntactic restrictions. For example, as discussed earlier, CSP(K_3), which is the same as 3-Colorability, is expressible by the sentence $\exists B \exists R \exists G \forall x \forall y\, \theta$, where θ is the quantifier-free formula

$$(B(x) \vee R(x) \vee G(x)) \wedge \neg(B(x) \wedge R(x)) \wedge \neg(B(x) \wedge G(x)) \wedge \neg(R(x) \wedge G(x)) \\ \wedge \big(\neg E(x,y) \vee (\neg(B(x) \wedge B(y)) \wedge \neg(R(x) \wedge R(y)) \wedge \neg(G(x) \wedge G(y)))\big).$$

Similarly, Positive 1-in-3 Sat is expressible by the sentence $\exists S \forall x \forall y \forall z\, \eta$, where η is the formula

$$\neg R(x,y,z) \vee (S(x) \wedge \neg S(y) \wedge \neg S(z)) \vee (\neg S(x) \wedge S(y) \wedge \neg S(z)) \vee (\neg S(x) \wedge \neg S(y) \wedge S(z)).$$

The preceding sentences of existential second-order logic obey the following syntactic restrictions: (a) all second-order quantifiers are monadic; (b) all first-order quantifiers are universal; (c) no inequalities occur; (d) all occurrences of relation symbols from the underlying vocabulary τ are preceded by the negation symbol. Taken together, these syntactic restrictions define the fragment of existential second-order logic known as MMSNP.

MMSNP has strictly higher expressive power than constraint satisfaction, in the sense that there are problems that are definable by a MMSNP-sentence, but are not expressible as a CSP($\mathfrak{B}$) problem for any structure $\mathfrak{B}$ over the same vocabulary. Indeed, as pointed out in [15], the problem "given a graph, is it triangle-free?" is expressible by the sentence

$$\forall x \forall y \forall z (\neg E(x,y) \vee \neg E(x,z) \vee \neg E(y,z)),$$

which is in the first-order part of MMSNP, but there is no graph H such that a graph G is triangle-free if and only if there is a homomorphism from G to H. Towards a contradiction, assume that such a graph H exists. Erdös [3] showed that there are graphs of arbitrarily large girth and chromatic number. It follows that there is a graph G that is triangle-free (i.e., G has girth at least 4) and chromatic number bigger than that of H. Thus, G is triangle-free, but there is no homomorphism from G to H, else we could color G with at most the number of colors needed to color H.

As mentioned in the Introduction, however, Feder and Vardi [5] showed that every MMSNP-definable problem is equivalent under polynomial-time reductions to a constraint satisfaction problem CSP($\mathfrak{B}$), for some structure $\mathfrak{B}$ over the same vocabulary. Consequently, establishing a dichotomy theorem for the complexity of model checking MMSNP-sentences is precisely as hard as affirming the Feder-Vardi dichotomy conjecture for constraint satisfaction.

2.2 Dependence logic

Dependence logic D is the extension of first-order logic augmented with dependence atoms $\mathrm{dep}(x_1, \ldots, x_n; y)$. Since dependence atoms are allowed to occur only positively in formulas of D, it is natural assume that all formulas are in negation normal form. Thus, we define the syntax of D by the following grammar:

$$\begin{aligned} \varphi \quad ::= \quad & x_1 = x_2 \mid \neg\, x_1 = x_2 \mid R(x_1, \ldots, x_n) \mid \neg R(x_1, \ldots, x_n) \mid \\ & \mathrm{dep}(x_1, \ldots, x_n; y) \mid (\varphi_1 \wedge \varphi_1) \mid (\varphi_1 \vee \varphi_2) \mid \forall x \varphi \mid \exists x \varphi. \end{aligned}$$

The semantics of D is defined with respect to *teams*, i.e., sets of assignments, instead of single assignments. If $\mathfrak{A}$ is a structure with domain A and V is a set of first-order variables, then an *assignment* on $\mathfrak{A}$ is a function $s : V \to A$. A *team* on $\mathfrak{A}$ is a set T of assignments on some fixed set $V = \mathrm{dom}(T)$ of variables. In particular, if $V = \emptyset$, then there are two teams on $\mathfrak{A}$ with domain V: the empty team $\emptyset$, and the team $T = \{\emptyset\}$ consisting of the empty assignment $\emptyset : \emptyset \to A$.

To define the semantics of universal quantification, we use the following notation: $T[A/x] = \{s[a/x] \mid s \in T, a \in A\}$, where $s[a/x]$ is the assignment such that it agrees with s on all $y \in \mathrm{dom}(s) \setminus \{x\}$, and $s[a/x](x) = a$.

To define the semantics for existential quantification, we need the notion of a *choice function* $F : T \to A$. The idea is that F picks an element $F(s)$ from the domain A of a structure $\mathfrak{A}$ for each assignment s in a team T. The element $F(s)$ is then used to interpret a variable x, thus obtaining the new assignment $s[F(s)/x]$. We write $T[F/x]$ for the team $\{s[F(s)/x] \mid s \in T\}$ obtained from T by making this change to each $s \in T$.

▶ **Definition 1.** Let $\mathfrak{A}$ be a model and T a team on $\mathfrak{A}$. The truth relation $\mathfrak{A}, T \models \varphi$ for dependence logic is defined as follows.

$$\begin{aligned}
&\mathfrak{A}, T \models x_1 = x_2 && \iff && s(x_1) = s(x_2) \text{ for all } s \in T.\\
&\mathfrak{A}, T \models \neg x_1 = x_2 && \iff && s(x_1) \neq s(x_2) \text{ for all } s \in T.\\
&\mathfrak{A}, T \models R(x_1, \ldots, x_n) && \iff && (s(x_1), \ldots, s(x_n)) \in R^{\mathfrak{A}} \text{ for all } s \in T.\\
&\mathfrak{A}, T \models \neg R(x_1, \ldots, x_n) && \iff && (s(x_1), \ldots, s(x_n)) \notin R^{\mathfrak{A}} \text{ for all } s \in T.\\
&\mathfrak{A}, T \models \mathrm{dep}(x_1, \ldots, x_n; y) && \iff && \text{there is a function } f : A^n \to A \text{ such that}\\
& && && s(y) = f(s(x_1), \ldots, s(x_n)) \text{ for all } s \in T.\\
&\mathfrak{A}, T \models \varphi \wedge \psi && \iff && \mathfrak{A}, T \models \varphi \text{ and } \mathfrak{A}, T \models \psi.\\
&\mathfrak{A}, T \models \varphi \vee \psi && \iff && \text{there are } T', T'' \subseteq T \text{ such that } T \cup T' = T'',\\
& && && \mathfrak{A}, T' \models \varphi \text{ and } \mathfrak{A}, T'' \models \psi.\\
&\mathfrak{A}, T \models \forall x \psi && \iff && \mathfrak{A}, T[A/x] \models \psi.\\
&\mathfrak{A}, T \models \exists x \psi && \iff && \text{there is a function } F \colon T \to A \text{ s.t. } \mathfrak{A}, T[F/x] \models \psi.
\end{aligned}$$

The set $\mathrm{Fr}(\varphi)$ of free variables of a formula $\varphi \in \mathsf{D}$ is defined in the standard way. The formula φ is a *sentence* if $\mathrm{Fr}(\varphi) = \emptyset$. A sentence $\varphi \in \mathsf{D}$ is *true* in a structure $\mathfrak{A}$, in symbols $\mathfrak{A} \models \varphi$, if $\mathfrak{A}, \{\emptyset\} \models \varphi$.

Note that in the literature (see, e.g., [17]), the semantics of the dependence atom is usually stated in the following equivalent form:

$$\mathfrak{A}, T \models \mathrm{dep}(x_1, \ldots, x_n; y) \iff \text{for all } s, s' \in T, \text{ if } s(x_i) = s'(x_i) \text{ for all } i \in \{1, \ldots, n\}, \text{ then } s(y) = s'(y).$$

Note also that, in database terminology, $\mathfrak{A}, T \models \mathrm{dep}(x_1, \ldots, x_n; y)$ means that the team T, viewed as an n-ary relation, satisfies the functional dependency $x_1, \ldots, x_n \to y$.

We review here briefly the basic properties of dependence logic. The first property is that the team semantics for first-order formulas in D (i.e., formulas without dependence atoms) can be reduced to the standard Tarski semantics. We write $\mathfrak{A}, s \models \varphi$ if the first-order formula φ is satisfied by the assignment s in the structure $\mathfrak{A}$.

▶ **Fact 1** (Flatness, [17]). *Let φ be a formula of D without dependence atoms, and let $\mathfrak{A}$ be a structure and T a team on $\mathfrak{A}$. Then $\mathfrak{A}, T \models \varphi$ if and only if $\mathfrak{A}, s \models \varphi$ for all $s \in T$.*

The second property is that the semantics of every D-formula is downwards closed in the following sense.

▶ **Fact 2** (Downward closure, [17]). *Let φ be a formula of D. If T and T' are teams on a structure $\mathfrak{A}$ such that $\mathfrak{A}, T \models \varphi$ and $T' \subseteq T$, then $\mathfrak{A}, T' \models \varphi$.*

The formulas φ of D also have the desirable property that the truth of φ only depends on the interpretation of its free variables $\mathrm{Fr}(\varphi)$. We use here the notation $T \restriction V = \{s \restriction V \mid s \in T\}$ for a team T and a set V of variables.

▶ **Fact 3** (Locality, [17]). *Let φ be a formula of D with $\mathrm{Fr}(\varphi) = V$. If T is a team on a structure $\mathfrak{A}$ and $T' = T \restriction V$, then $\mathfrak{A}, T \models \varphi$ if and only if $\mathfrak{A}, T' \models \varphi$.*

Finally, as mentioned in the Introduction, dependence logic D has the same expressive power as existential second-order logic Σ^1_1.

▶ **Fact 4** (D captures Σ^1_1, [17]). *For every sentence φ of D, there is an equivalent sentence ψ of Σ^1_1; vice versa, for every sentence ψ of Σ^1_1, there is an equivalent sentence φ of D.*

As a consequence of Fact 4 and Fagin's Theorem [4], dependence logic D captures the complexity class NP. In particular, this means that NP-complete problems, such as k-Colorability and k-Sat, $k \geq 3$, are expressible in D. Perhaps surprisingly, it turns out that the model-checking problem of D-formulas can be NP-complete already at the quantifier-free level. Specifically, Jarmo Kontinen [12] proved that the problem "does a team T on a structure $\mathfrak{A}$ (with empty vocabulary) satisfy the formula $\mathrm{dep}(x; y) \vee \mathrm{dep}(u; v) \vee \mathrm{dep}(u; v)$?" is NP-complete. On the other hand, he proved that the model-checking problem for the disjunction of any two dependence atoms is in NLOGSPACE.

The complexity of model-checking for quantifier-free formulas of D has been further investigated by Durand et al. [2]. Extending the ideas of Kontinen [12], they give sufficient syntactic criteria for the tractability and the NP-completeness of such model-checking problems. In the present paper, we focus on the relationship between the universal fragment of dependence logic, constraint satisfaction problems and MMSNP, and unveil a tight connection.

2.3 Logics with k-valued variables

In the next subsection, we will define uniform k-valued dependence atoms. To do this, in addition to the usual first-order variables, we need a separate supply of k-valued variables. Furthermore, to interpret the k-valued variables, we will extend structures by a standard part consisting of the numbers $1, \dots, k$. Thus, if $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_n^{\mathfrak{A}})$ is a τ-structure, then we define $\mathfrak{A}[k]$ to be the two-sorted structure $(\mathfrak{A}; [k], \underline{1}^{\mathfrak{A}}, \dots, \underline{k}^{\mathfrak{A}})$. Here $[k]$ is the domain of the second sort and $\underline{1}, \dots, \underline{k}$ are constant symbols over the second sort such that $\underline{i}^{\mathfrak{A}} = i$ for each $i \in [k]$.

We will use the Greek letters α, β, γ, with or without subscripts, as *k-valued* variables, while we will use x, y, u, v as ordinary *first-order* variables. The intuition is that k-valued variables always range over the second sort $[k]$ of a structure $\mathfrak{A}[k]$, while the first-order variables range over the domain A of $\mathfrak{A}$. We often use the boldface notation $\boldsymbol{x}$ ($\boldsymbol{\alpha}$, or $\boldsymbol{a}$) for a tuple $(x_1, \dots, x_n)$ of variables (a tuple $(\alpha_1, \dots, \alpha_n)$ of k-valued variables, or a tuple $(a_1, \dots, a_n)$ of elements, respectively). If not explicitly defined, the length n of the tuple will be clear from the context.

For logics with k-valued variables and team semantics, the notion of a team needs to be adapted. If $\mathfrak{A}$ is a structure, and V is a finite set of first-order and k-valued variables, then an assignment on $\mathfrak{A}[k]$ with domain V is a function $s : V \to A \cup [k]$ such that $s(x) \in A$ for each first-order variable $x \in V$ and $s(\alpha) \in [k]$ for each k-valued variable $\alpha \in V$. A *team* on $\mathfrak{A}[k]$ with domain V is a set T of assignments $s : V \to A \cup [k]$.

We will next introduce some useful notation.

▶ **Definition 2.** Let T be a team on a structure $\mathfrak{A}[k]$ with domain V.

- If $\boldsymbol{x} \in V^n$ and $\boldsymbol{\alpha} \in V^m$, then we use the notation $R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$ for the $(n+m)$-ary relation $\{s(\boldsymbol{x}\boldsymbol{\alpha}) \mid s \in T\} \subseteq A^n \times [k]^m$.
- In case $m = 0$, we write simply $R_{T,\boldsymbol{x}} = \{s(\boldsymbol{x}) \mid s \in T\}$. Similarly, in case $n = 0$, we write $R_{T,\boldsymbol{\alpha}} = \{s(\boldsymbol{\alpha}) \mid s \in T\}$.
- Furthermore, if $\boldsymbol{a} \in A^n$, then $T[\boldsymbol{x} = \boldsymbol{a}]$ denotes the subteam $\{s \in T \mid s(\boldsymbol{x}) = \boldsymbol{a}\} \subseteq T$. Similarly, if $\boldsymbol{\ell} \in [k]^m$, then $T[\boldsymbol{\alpha} = \boldsymbol{\ell}]$ denotes the subteam $\{s \in T \mid s(\boldsymbol{\alpha}) = \boldsymbol{\ell}\} \subseteq T$.

Note that, in database terminology, $R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$ is the *projection* $\pi_{\boldsymbol{x}\boldsymbol{\alpha}}(T)$ of the team T on the variables $\boldsymbol{x}\boldsymbol{\alpha}$, where T is viewed as a relation. Moreover, $T[\boldsymbol{x} = \boldsymbol{a}]$ is the *selection* $\sigma_{\boldsymbol{x}=\boldsymbol{a}}(T)$ of the team T, where T is viewed as a relation; similarly, $T[\boldsymbol{\alpha} = \boldsymbol{\ell}]$ is the selection $\sigma_{\boldsymbol{\alpha}=\boldsymbol{\ell}}(T)$.

To simplify the notation, henceforth we will denote the structures $\mathfrak{A}[k]$ simply by $\mathfrak{A}$. This should not cause any confusion, since it is always clear from the context, whether the symbol $\mathfrak{A}$ refers to a usual structure, or the extension of such structure with the second sort $[k]$.

2.4 Uniform k-valued dependence atoms

We are now ready to define the *uniform k-valued dependence atoms*, which we will use in the rest of the paper. These atoms differ from the standard dependence atoms in two ways: first, they are k-valued; second, the functional dependence is generated by a single unary function.

▶ **Definition 3.** If $\boldsymbol{x} = (x_1, \dots, x_n)$ is an n-tuple of first-order variables and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ is an n-tuple of k-valued variables, then $\mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha})$ is an atomic formula with the semantics

$$\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha}) \iff \begin{array}{l}\text{there is a function } f : A \to [k] \text{ such that} \\ s(\alpha_i) = f(s(x_i)), \text{ for all } i \in [n] \text{ and } s \in T.\end{array}$$

Note that in the case $n = 1$, the uniform k-valued dependence atom $\mathrm{udep}[k](x; \alpha)$ is equivalent with the k-valued version $\mathrm{dep}[k](x; \alpha)$ of the ordinary dependence atom $\mathrm{dep}(x; y)$.

The semantics of universal and existential quantification of k-valued variables can be defined in the same way as for quantification of first-order variables by defining $T[[k]/\alpha] = \{s[i/\alpha] \mid s \in T, i \in [k]\}$, and $T[G/\alpha] = \{s[G(s)/\alpha] \mid s \in T\}$ for a choice function $G : T \to [k]$. However, we will not consider existential quantification in this paper, as our main focus is on a quantifier-free fragment of the full logic with uniform k-valued dependence atoms, and its closure with respect to universal quantifiers.

▶ **Definition 4.** The *quantifier-free monotone dependence logic with uniform k-valued dependence atoms*, $\mathsf{QF\text{-}MUD}[k]$, is defined by the following grammar:

$$\varphi \quad ::= \quad \alpha = \underline{i} \mid \neg R(\boldsymbol{x}) \mid \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha}) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2), \quad \text{where } i \in [k].$$

Universal monotone dependence logic with uniform k-valued dependence atoms, $\forall\mathsf{\text{-}MUD}[k]$, is the extension of $\mathsf{QF\text{-}MUD}[k]$ defined by the grammar

$$\varphi \quad ::= \quad \psi \mid \forall x\varphi \mid \forall\alpha\varphi, \quad \text{where } \psi \in \mathsf{QF\text{-}MUD}[k].$$

The union of $\forall\mathsf{\text{-}MUD}[k]$ over all $k \geq 1$ is denoted by $\forall\mathsf{\text{-}MUD}[\omega]$. Similarly, $\mathsf{QF\text{-}MUD}[\omega]$ is the union of $\mathsf{QF\text{-}MUD}[k]$ over all $k \geq 1$.

Thus, analogously to MMSNP, the logics $\mathsf{QF\text{-}MUD}[k]$ and $\forall\mathsf{\text{-}MUD}[k]$ admit no inequalities and only negative occurrences of relation symbols in the vocabulary. Note that there is no need to include equalities of the form $\alpha = \beta$, since they can be expressed as $\bigvee_{i\in[k]}(\alpha = \underline{i} \wedge \beta = \underline{i})$. Furthermore, inequalities between k-valued variables are also expressible: $\alpha \neq \beta$ is equivalent to $\bigvee_{i\in[k]}\big(\alpha = \underline{i} \wedge \bigvee_{j\in[k],j\neq i} \beta = \underline{j}\big)$.

For the sake of completeness, we state here the definition of the semantics of $\forall\mathsf{\text{-}MUD}[k]$.

▶ **Definition 5.** Let $\mathfrak{A}$ be a structure and T a team on $\mathfrak{A}$. The *truth* relation $\mathfrak{A}, T \models \varphi$ for universal monotone uniform k-valued dependence logic is defined as follows.

$$\begin{array}{lcl}
\mathfrak{A}, T \models \alpha = \underline{i} & \iff & s(\alpha) = i \text{ for all } s \in T. \\
\mathfrak{A}, T \models \neg R(\boldsymbol{x}) & \iff & (s(x_1), \ldots, s(x_n)) \notin R^{\mathfrak{A}} \text{ for all } s \in T. \\
\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha}) & \iff & \text{there is a function } f : A \to [k] \text{ such that} \\
 & & s(\alpha_i) = f(s(x_i)) \text{ for all } i \in [n] \text{ and } s \in T. \\
\mathfrak{A}, T \models \varphi \wedge \psi & \iff & \mathfrak{A}, T \models \varphi \text{ and } \mathfrak{A}, T \models \psi. \\
\mathfrak{A}, T \models \varphi \vee \psi & \iff & \text{there are } T', T'' \subseteq T \text{ such that } T' \cup T'' = T, \\
 & & \mathfrak{A}, T' \models \varphi \text{ and } \mathfrak{A}, T'' \models \psi. \\
\mathfrak{A}, T \models \forall x\varphi & \iff & \mathfrak{A}, T[A/x] \models \varphi. \\
\mathfrak{A}, T \models \forall\alpha\varphi & \iff & \mathfrak{A}, T[[k]/\alpha] \models \varphi.
\end{array}$$

Since dependence logic has the same expressive power as existential second-order logic, it is clear that uniform k-valued dependence atoms are definable in D (in the setting with k-valued variables). Indeed, it is straightforward to check that $\mathrm{udep}[k](x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n)$ is equivalent to the formula

$$\forall y \exists \beta \big(\mathrm{dep}[k](y;\beta) \wedge \bigwedge_{i\in[n]} (y = x_i \to \beta = \alpha_i)\big).$$

Note however, that this formula violates the syntactic restrictions of $\forall\mathsf{\text{-}MUD}[k]$ in two different ways: it contains existential quantification of a k-valued variable and inequalities between first-order variables.

As in the case of dependence logic D, a formula φ of $\forall\mathsf{\text{-}MUD}[k]$ is a sentence, if the set $\mathrm{Fr}(\varphi)$ of its free variables is empty. Furthermore, a sentence φ is true in a structure $\mathfrak{A}$, in symbols $\mathfrak{A} \models \varphi$, if $\mathfrak{A}, \{\emptyset\} \models \varphi$.

Clearly any $\forall$-MUD$[k]$-sentence φ is equivalent to a sentence of the form $\forall \boldsymbol{x} \forall \boldsymbol{\alpha} \psi$, where ψ is a QF-MUD$[k]$-formula. As a matter of fact, we can assume without loss of generality that φ is the *universal closure* of ψ, i.e., the tuple $\boldsymbol{x}\boldsymbol{\alpha}$ is repetition-free and consists of the free variables of ψ. Using the truth conditions for universal quantification of first-order and k-valued variables repeatedly, we obtain the following simple connection between the semantics of φ and ψ:

- $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{A}, F \models \psi$, where F is the team consisting of all assignments $s : V \to A \cup [k]$ with $V = \mathrm{Fr}(\psi)$.

We will call F the *full team* (on $\mathfrak{A}$ with domain V) in the sequel. If there is need to emphasize the domain V of F, we denote the full team by F_V.

The full team has a special role in the semantics of QF-MUD$[k]$ also in another way. It is straightforward to verify that Facts 2 and 3 (see Subsection 2.2) remain true for $\forall$-MUD$[k]$. Specifically, for every formula $\psi \in$ QF-MUD$[k]$, the following statements are true:

1. if $\mathfrak{A}, T \models \psi$ and $T' \subseteq T$, then $\mathfrak{A}, T' \models \psi$.
2. if $T' = T \upharpoonright \mathrm{Fr}(\psi)$, then $\mathfrak{A}, T \models \psi$ if and only if $\mathfrak{A}, T' \models \psi$.

Thus, to decide whether a formula is satisfied by every team in a given structure, it suffices to check whether it is satisfied by the full team.

We summarize the two observations concerning the full team in the following lemma.

▶ **Lemma 6.** *Let ψ be a* QF-MUD$[k]$*-formula with $\boldsymbol{x}$ and $\boldsymbol{\alpha}$ as its free variables. Then the following statements are equivalent:*

1. $\mathfrak{A} \models \forall \boldsymbol{x} \forall \boldsymbol{\alpha} \psi$.
2. $\mathfrak{A}, F \models \psi$.
3. $\mathfrak{A}, T \models \psi$, *for every team T on $\mathfrak{A}$ with* $\mathrm{Fr}(\psi) \subseteq \mathrm{dom}(T)$

3 From Constraint Satisfaction to Dependence Logic

Our aim in this section is to prove that every constraint satisfaction problem CSP($\mathfrak{B}$) is captured by a sentence of $\forall$-MUD$[\omega]$. To do this, we will prove that CSP($\mathfrak{B}$) is definable in $\forall$-MUD$[\omega]$, assuming that $\mathfrak{B}$ is of the form $(B, R^{\mathfrak{B}})$, i.e., $\mathfrak{B}$ has only one relation. This suffices, since as mentioned in Subsection 2.1, every constraint satisfaction problem CSP($\mathfrak{B}$) is equivalent, via polynomial-time reductions, to a CSP($\mathfrak{B}'$) in which $\mathfrak{B}'$ is a structure with a single binary relation.

We start by observing that the truth of a $[k]$-valued uniform dependence atom on a given structure $\mathfrak{A}$ and a given team T implies the existence of a homomorphism between the two structures $(A, R_{T,\boldsymbol{x}})$ and $([k], R_{T,\boldsymbol{\alpha}})$.

▶ **Lemma 7.** *If $\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha})$, then $(A, R_{T,\boldsymbol{x}}) \in \mathrm{CSP}([k], R_{T,\boldsymbol{\alpha}})$.*

Proof. Assume that $\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha})$. Then there is a function $f : A \to [k]$ such that

- $f(s(x_i)) = s(\alpha_i)$ for all $i \in [n]$ and $s \in T$.

This condition implies that f is a homomorphism from $(A, R_{T,\boldsymbol{x}})$ to $([k], R_{T,\boldsymbol{\alpha}})$. Indeed, if $\boldsymbol{a} = (a_1, \ldots, a_n) \in R_{T,\boldsymbol{x}}$, then there exists $s \in T$ such that $s(x_i) = a_i$ for all $i \in [n]$. But then also $s(\alpha_i) = f(a_i)$ holds for all $i \in [n]$, whence $(f(a_1), \ldots, f(a_n)) \in R_{T,\boldsymbol{\alpha}}$. ◀

Note that the converse implication of Lemma 7 is not true. As an example, consider the team $T = \{s, s'\}$, where $s(x_1) = s'(x_1)$, $s(x_2) = s'(x_2)$, $s(\alpha_1) = s(\alpha_2) = 1$ and $s'(\alpha_1) = s'(\alpha_2) = 2$. Then the function $h : A \to [k]$ such that $h(a) = 1$ for all $a \in A$, is a homomorphism $(A, R_{T,x_1x_2}) \to ([k], R_{T,\alpha_1\alpha_2})$, but clearly $\mathfrak{A}, T \not\models \mathrm{udep}[k](x_1, x_2; \alpha_1, \alpha_2)$. Thus, uniform dependence atoms are different from *homomorphism* atoms.

For each positive integer n, let τ_n be a vocabulary consisting of a single n-ary relation symbol R. Let $\mathfrak{B} = ([k], R^{\mathfrak{B}})$ be a τ_n-structure, where $R^{\mathfrak{B}} \neq \emptyset$. The crucial step in the proof that CSP($\mathfrak{B}$) is expressible in $\forall$-MUD$[k]$, is the following. Assume that $\mathfrak{A}$ is a τ_n-structure and T is a team on $\mathfrak{A}$ such that $\boldsymbol{a}\boldsymbol{\ell} \in R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$ for all $\boldsymbol{a} \in R^{\mathfrak{A}}$ and $\boldsymbol{\ell} \in R^{\mathfrak{B}}$ and $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$. We define a formula $\theta_{R^{\mathfrak{B}}} \in$ QF-MUD$[k]$ that is true in a subteam T' of T if and only if for each $\boldsymbol{a} \in R^{\mathfrak{A}}$, $R_{T'[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}}$ contains all except exactly one of the tuples $\boldsymbol{\ell} \in R^{\mathfrak{B}}$. If the complement $T'' = T \setminus T'$ of T' satisfies the dependence atom udep$[k](\boldsymbol{x};\boldsymbol{\alpha})$, it follows then from Lemma 7, that there is a homomorphism $\mathfrak{A} \to \mathfrak{B}$. Conversely, if there is a homomorphism $\mathfrak{A} \to \mathfrak{B}$, then T can be divided into subteams T' and T'' that satisfy $\theta_{R^{\mathfrak{B}}}$ and udep$[k](\boldsymbol{x};\boldsymbol{\alpha})$, respectively. The precise statement and proof of this equivalence is given in Lemma 9, below.

Before defining $\theta_{R^{\mathfrak{B}}}$, we introduce some useful auxiliary formulas. If y and β are variables, we let $\chi_1(y,\beta) := \mathrm{udep}[k](y;\beta)$, and define recursively $\chi_{r+1}(y,\beta) := (\chi_r(y,\beta) \vee \chi_1(y,\beta))$. Thus, $\chi_r(y,\beta)$ is the disjunction of r copies of the dependence atom udep$[k](y;\beta)$ with itself. Furthermore, define $\chi_0(y,\beta)$ to be a formula that is always false (e.g., $\beta = \underline{1} \wedge \beta = \underline{2}$). It is easy to show, by induction on r, that

$$\mathfrak{A}, T \models \chi_r(y,\beta) \quad \text{if and only if } |R_{T[y=a],\beta}| \leq r, \text{ for every } a \in R_{T,y}.$$

Next, for each nonempty relation $S \subseteq [k]^n$, we define a formula θ_S by induction on the arity n of S:

1. For $n = 1$, we let $\theta_S := \chi_{s-1}(x_1, \alpha_1)$, where $s = |S|$.
2. Let S be a nonempty n-ary relation on $[k]$, $n > 1$. Then there is a unique set $I \subseteq [k]$ and unique nonempty relations $S_\ell \subseteq [k]^{n-1}$, $\ell \in I$, such that $S = \bigcup_{\ell \in I} S_\ell \times \{\ell\}$. We define

$$\theta_S := \chi_{m-1}(x_n, \alpha_n) \vee \bigvee_{\ell \in I} (\alpha_n = \underline{\ell} \wedge \theta_{S_\ell}), \quad \text{where } m = |I|.$$

To illustrate the preceding notions, consider the structure $\mathfrak{B} = (\{0,1\}, R^{\mathfrak{B}})$, where $R^{\mathfrak{B}} = \{(1,0,0),(0,1,0),(0,0,1)\}$. As mentioned in Section 2.1, the constraint satisfaction problem CSP($\mathfrak{B}$) amounts to the satisfiability problem Positive 1-in-3 Sat. We are interested in computing the formula $\theta_{R^{\mathfrak{B}}}$. Clearly, $R^{\mathfrak{B}} = \{(0,1),(1,0)\} \times \{0\} \cup \{(0,0)\} \times \{1\}$. Thus, by taking $I = \{0,1\}$, $S_0 = \{(0,1),(1,0)\}$, and $S_1 = \{(0,0)\}$, we have that

$$\theta_{R^{\mathfrak{B}}} := \chi_1(x_3,\alpha_3) \vee (\alpha_3 = \underline{0} \wedge \theta_{S_0}) \vee (\alpha_3 = \underline{1} \wedge \theta_{S_1}).$$

We leave it to the reader to verify that, after unraveling further the definitions and eliminating disjuncts containing formulas that are always false, we have that

$$\theta_{R^{\mathfrak{B}}} := \mathrm{udep}[2](x_3;\alpha_3) \vee (\alpha_3 = \underline{0} \wedge \mathrm{udep}[2](x_2;\alpha_2)).$$

▶ **Lemma 8.** *Let $S \subseteq [k]^n$ be a nonempty relation, and let T be a team on a structure $\mathfrak{A}$.*

1. *If $\mathfrak{A}, T \models \theta_S$, then for every $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$, we have that $S \setminus R_{T[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}} \neq \emptyset$.*
2. *If $R_{T,\boldsymbol{\alpha}} \subseteq S$ and $f : A \to [k]$ is a function such that $(f(a_1),\ldots,f(a_n)) \in S \setminus R_{T[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}}$ for all $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$, then $\mathfrak{A}, T \models \theta_S$.*

Proof. Both claims are proved by induction on n.

1. If $n = 1$ and $\mathfrak{A}, T \models \theta_S$, then by the definition of θ_S and the observation above, $|R_{T[x_1=a],\alpha_1}| \leq |S| - 1$ for all $a \in R_{T,x_1}$. Thus, $S \setminus R_{T[x_1=a],\alpha_1} \neq \emptyset$ for all $a \in R_{T,x_1}$. Assume then that $S = \bigcup_{\ell \in I} S_\ell \times \{\ell\}$ is a nonempty n-ary relation on $[k]$, and the claim is true for the relations $S_\ell \subseteq [k]^{n-1}$, $\ell \in I$. If $\mathfrak{A}, T \models \theta_S$, then there are subsets T_0 and T_ℓ, $\ell \in I$, such that

- $T = T_0 \cup \bigcup_{\ell \in I} T_\ell$;
- $\mathfrak{A}, T_0 \models \chi_{m-1}(x_n, \alpha_n)$ for $m = |I|$;
- $\mathfrak{A}, T_\ell \models \alpha_n = \underline{\ell} \wedge \theta_{S_\ell}$ for each $\ell \in I$.

Let $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$. By the second clause above, $|R_{T_0[x_n = a_n],\alpha_n}| < |I|$, whence there exists $\ell \in I$ such that $\ell \notin R_{T_0[x_n = a_n],\alpha_n}$. Since $\mathfrak{A}, T_\ell \models \theta_{S_\ell}$, by induction hypothesis there exists a tuple $\boldsymbol{\ell}^* \in S_\ell \setminus R_{T_\ell[\boldsymbol{x}^* = \boldsymbol{a}^*],\boldsymbol{\alpha}^*}$, where $\boldsymbol{a}^* = (a_1, \ldots, a_{n-1})$, $\boldsymbol{x}^* = (x_1, \ldots, x_{n-1})$ and $\boldsymbol{\alpha}^* = (\alpha_1, \ldots, \alpha_{n-1})$.

Consider now the tuple $\boldsymbol{\ell} = \boldsymbol{\ell}^*\ell$. Since $\boldsymbol{\ell}^* \in S_\ell$ and $S_\ell \times \{\ell\} \subseteq S$, we have $\boldsymbol{\ell} \in S$. On the other hand, $\boldsymbol{\ell} \notin R_{T_0[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$, since $\ell \notin R_{T_0[x_n = a_n],\alpha_n}$. Furthermore, $\mathfrak{A}, T_j \models \alpha_n = \underline{j}$, whence $\boldsymbol{\ell} \notin R_{T_j[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$ for all $j \in I \setminus \{\ell\}$. Finally, $\boldsymbol{\ell} \notin R_{T_\ell[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$, since $\boldsymbol{\ell}^* \in S_\ell \setminus R_{T_\ell[\boldsymbol{x}^* = \boldsymbol{a}^*],\boldsymbol{\alpha}^*}$. Thus, we conclude that $\boldsymbol{\ell} \notin R_{T[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$, as $T = T_0 \cup \bigcup_{\ell \in I} T_\ell$.

2. Assume that $R_{T,\boldsymbol{\alpha}} \subseteq S$ and $f : A \to [k]$ is a function satisfying the condition

$$(f(a_1), \ldots, f(a_n)) \in S \setminus R_{T[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}} \quad \text{for all } \boldsymbol{a} \in R_{T,\boldsymbol{x}}.$$

If $n = 1$, this implies that $R_{T[x_1 = a_1],\alpha_1} \subseteq R_{T,\alpha_1} \setminus \{f(a_1)\} \subseteq S \setminus \{f(a_1)\}$, whence $|R_{T[x_1 = a_1],\alpha_1}| \leq |S| - 1$ for all $a_1 \in R_{T,x_1}$. Thus, $\mathfrak{A}, T \models \theta_S$ in the case $n = 1$.

Assume then that $S = \bigcup_{\ell \in I} S_\ell \times \{\ell\}$ is a nonempty n-ary relation on $[k]$, and the claim is true for the relations $S_\ell \subseteq [k]^{n-1}$, $\ell \in I$. We define subteams T_0 and T_ℓ, $\ell \in I$, as follows:

- $T_0 := \{s \in T \mid s(\alpha_n) \neq f(s(x_n))\}$, and
- $T_\ell := \{s \in T \mid s(\alpha_n) = f(s(x_n)) = \ell\}$, for $\ell \in I$.

It is clear from these definitions that $T = T_0 \cup \bigcup_{\ell \in I} T_\ell$.

Since $R_{T_0[x_n = a_n],\alpha_n} \subseteq R_{T,\alpha_n} \setminus \{f(a_n)\} \subseteq I \setminus \{f(a_n)\}$, we have $|R_{T_0[x_n = a_n],\alpha_n}| \leq |I| - 1$, and hence $\mathfrak{A}, T_0 \models \chi_{m-1}(x_n, \alpha_n)$ for $m = |I|$. Furthermore, it follows from the definition of T_ℓ that $\mathfrak{A}, T_\ell \models \alpha_n = \underline{\ell}$ for each $\ell \in I$.

It remains to prove that $\mathfrak{A}, T_\ell \models \theta_{S_\ell}$ for each $\ell \in I$. Assume for this purpose that $\boldsymbol{a}^* \in R_{T_\ell,\boldsymbol{x}^*}$, where $\boldsymbol{x}^*$ is the initial segment $(x_1, \ldots, x_{n-1})$ of $\boldsymbol{x}$. Then there is an extension $\boldsymbol{a}$ of $\boldsymbol{a}^*$ by an n-th component a_n such that $\boldsymbol{a} \in R_{T_\ell,\boldsymbol{x}}$. By the assumption on f, we have $(f(a_1), \ldots, f(a_n)) \in S \setminus R_{T[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}} \subseteq S \setminus R_{T_\ell[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$. On the other hand, by the definition of T_ℓ, we have $f(a_n) = \ell \in R_{T_\ell[x_n = a_n],\alpha_n}$, whence it must be the case that $(f(a_1), \ldots, f(a_{n-1})) \in S_\ell \setminus R_{T_\ell[\boldsymbol{x}^* = \boldsymbol{a}^*],\boldsymbol{\alpha}^*}$. Since this holds for every $\boldsymbol{a}^* \in R_{T_\ell,\boldsymbol{x}^*}$, it follows from the induction hypothesis that $\mathfrak{A}, T_\ell \models \theta_{S_\ell}$. ◀

We can now define a formula $\eta_{\mathfrak{B}}$ of QF-MUD$[k]$ with free variables $x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_n$ that expresses CSP$(\mathfrak{B})$ on teams that satisfy the two conditions mentioned in the discussion after Lemma 7. The formula is defined as follows:

$$\eta_{\mathfrak{B}} := \theta_{R^{\mathfrak{B}}} \vee \mathrm{udep}[k](x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n).$$

▶ **Lemma 9.** *Let $\mathfrak{A}$ be a τ_n-structure and let T be a team on A such that $R^{\mathfrak{A}} \times R^{\mathfrak{B}} \subseteq R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$ and $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$. Then the following statements are equivalent:*

1. $\mathfrak{A} \in \mathrm{CSP}(\mathfrak{B})$.
2. $\mathfrak{A}, T \models \eta_{\mathfrak{B}}$.

Proof. Assume first that h is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. We define two subteams T'' and T' of T as follows:

$$T'' := \{s \in T \mid s(\alpha_i) = h(s(x_i)) \text{ for all } i \in [n]\} \quad \text{and} \quad T' = T \setminus T''.$$

Clearly, $(h(a_1), \ldots, h(a_n)) \in R^{\mathfrak{B}} \setminus R_{T'[\boldsymbol{x} = \boldsymbol{a}],\boldsymbol{\alpha}}$, for all $\boldsymbol{a} \in R_{T',\boldsymbol{x}}$, whence by Lemma 8.2, $\mathfrak{A}, T' \models \theta_{R^{\mathfrak{B}}}$. Furthermore, $\mathfrak{A}, T'' \models \mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha})$ by the definition of T''. Thus, $\mathfrak{A}, T \models \eta_{\mathfrak{B}}$.

For the other direction, assume that $\mathfrak{A}, T \models \eta_{\mathfrak{B}}$. Then there are subteams T' and T'' of T such that $T = T' \cup T''$, $\mathfrak{A}, T' \models \theta_{\mathfrak{B}}$, and $A, T'' \models \mathrm{udep}[k](\boldsymbol{x}; \boldsymbol{\alpha})$. By Lemma 7, there is a homomorphism $h : (A, R_{T'',\boldsymbol{x}}) \to ([k], R_{T'',\boldsymbol{\alpha}})$. Since $T'' \subseteq T$, we have $R_{T'',\boldsymbol{\alpha}} \subseteq R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$. Furthermore, by Lemma 8.1, $R^{\mathfrak{B}} \setminus R_{T'[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}} \neq \emptyset$, for every $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$. On the other hand, by the assumption $R^{\mathfrak{A}} \times R^{\mathfrak{B}} \subseteq R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$, we have $R^{\mathfrak{B}} \setminus R_{T[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}} = \emptyset$, for each $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$. Thus, $R_{T''[\boldsymbol{x}=\boldsymbol{a}],\boldsymbol{\alpha}} \neq \emptyset$, for every $\boldsymbol{a} \in R_{T,\boldsymbol{x}}$, whence $R^{\mathfrak{A}} \subseteq R_{T,\boldsymbol{x}} = R_{T'',\boldsymbol{x}}$. As $R_{T'',\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$ and $R^{\mathfrak{A}} \subseteq R_{T'',\boldsymbol{x}}$, we conclude that h is a homomorphism $\mathfrak{A} \to \mathfrak{B}$. ◀

We observe next that for each τ_n-structure $\mathfrak{B} = ([k], R^{\mathfrak{B}})$, the properties $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$ and $R_{T,\boldsymbol{\alpha}} \cap R^{\mathfrak{B}} = \emptyset$ of teams are definable in $\mathsf{QF\text{-}MUD}[k]$. Indeed, if

$$\psi_{\mathfrak{B}} := \bigvee_{\boldsymbol{\ell} \in R^{\mathfrak{B}}} \Big(\bigwedge_{i \in [n]} \alpha_i = \underline{\ell}_i \Big) \qquad \text{and} \qquad \nu_{\mathfrak{B}} := \bigvee_{\boldsymbol{\ell} \notin R^{\mathfrak{B}}} \Big(\bigwedge_{i \in [n]} \alpha_i = \underline{\ell}_i \Big),$$

then clearly for any structure $\mathfrak{A}$ and team T on A, $\mathfrak{A}, T \models \psi_{\mathfrak{B}}$ if and only if $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$, and $\mathfrak{A}, T \models \nu_{\mathfrak{B}}$ if and only if $R_{T,\boldsymbol{\alpha}} \cap R^{\mathfrak{B}} = \emptyset$.

▶ **Theorem 10** ($\forall$-$\mathsf{MUD}[\omega]$ captures CSP)**.** *Let $\mathfrak{B} = ([k], R^{\mathfrak{B}})$ be a τ_n-structure. There is a sentence $\varphi_{\mathfrak{B}} \in \forall\text{-}\mathsf{MUD}[k]$ such that for every τ_n-structure $\mathfrak{A}$, the following statements are equivalent:*

1. $\mathfrak{A} \in \mathrm{CSP}(\mathfrak{B})$.
2. $\mathfrak{A} \models \varphi_{\mathfrak{B}}$.

Proof. Let $\xi_{\mathfrak{B}}$ be the $\mathsf{QF\text{-}MUD}[k]$-formula

$$(\eta_{\mathfrak{B}} \wedge \psi_{\mathfrak{B}}) \vee \neg R(x_1, \ldots, x_n) \vee \nu_{\mathfrak{B}}.$$

We define $\varphi_{\mathfrak{B}}$ to be the universal closure $\forall x_1 \ldots \forall x_n \forall \alpha_1 \ldots \forall \alpha_n \xi_{\mathfrak{B}}$ of the formula $\xi_{\mathfrak{B}}$. By Lemma 6, it suffices to prove that $\mathfrak{A} \in \mathrm{CSP}(\mathfrak{B})$ holds if and only if $\mathfrak{A}, F \models \xi_{\mathfrak{B}}$, where F is the full team on $\mathfrak{A}$ with domain $\{x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_n\}$.

Assume first that $\mathfrak{A} \in \mathrm{CSP}(\mathfrak{B})$. Let T' be the set of all assignments $s \in F$ such that $s(\boldsymbol{x}) \notin R^{\mathfrak{A}}$, and let T'' be the set of all assignments $s \in F$ such that $s(\boldsymbol{\alpha}) \notin R^{\mathfrak{B}}$. Then $\mathfrak{A}, T' \models \neg R(x_1, \ldots, x_n)$ and $\mathfrak{A}, T'' \models \nu_{\mathfrak{B}}$. Furthermore, $R^{\mathfrak{A}} \times R^{\mathfrak{B}} \subseteq R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$ and $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$, for $T = F \setminus (T' \cup T'')$. Thus, T satisfies the conditions of Lemma 9, whence $\mathfrak{A}, T \models \eta_{\mathfrak{B}}$. Since $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$, we also have $\mathfrak{A}, T \models \psi_{\mathfrak{B}}$. Since $F = T \cup T' \cup T''$, we conclude that $\mathfrak{A}, F \models \xi_{\mathfrak{B}}$.

For the other direction, assume that $\mathfrak{A}, F \models \xi_{\mathfrak{B}}$. Then there are subteams T, T' and T'' of F such that

- $F = T \cup T' \cup T''$;
- $\mathfrak{A}, T \models \eta_{\mathfrak{B}} \wedge \psi_{\mathfrak{B}}$;
- $\mathfrak{A}, T' \models \neg R(x_1, \ldots, x_n)$;
- $\mathfrak{A}, T'' \models \nu_{\mathfrak{B}}$.

If $s \in F$ is an assignment such that $s(\boldsymbol{x}) \in R^{\mathfrak{A}}$ and $s(\boldsymbol{\alpha}) \in R^{\mathfrak{B}}$, then it is not possible that $s \in T'$ or $s \in T''$, whence necessarily $s \in T$. Thus, we see that $R^{\mathfrak{A}} \times R^{\mathfrak{B}} \subseteq R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$. Furthermore, since $\mathfrak{A}, T \models \psi_{\mathfrak{B}}$, we have $R_{T,\boldsymbol{\alpha}} \subseteq R^{\mathfrak{B}}$. Hence, T satisfies the conditions of Lemma 9, and consequently $\mathfrak{A} \in \mathrm{CSP}(\mathfrak{B})$. ◀

We note that the size of the sentence $\varphi_{\mathfrak{B}}$ defining $\mathrm{CSP}(\mathfrak{B})$ is polynomial in the size of $\mathfrak{B}$. In fact, it is not hard to show that the size of $\varphi_{\mathfrak{B}}$ is $O(n \cdot k^{n+1})$, where n is the arity of $R^{\mathfrak{B}}$.

We illustrate the preceding theorem by considering again the structure $\mathfrak{B} = (\{0,1\}, R^{\mathfrak{B}})$, where $R^{\mathfrak{B}} = \{(1,0,0), (0,1,0), (0,0,1)\}$. In this case, we have that

$$\varphi_{\mathfrak{B}} := \forall x_1 \forall x_2 \forall x_3 \forall \alpha_1 \forall \alpha_2 \forall \alpha_3 \big((\eta_{\mathfrak{B}} \wedge \psi_{\mathfrak{B}}) \vee \neg R(x_1, x_2, x_3) \vee \nu_{\mathfrak{B}} \big),$$

where

- $\eta_{\mathfrak{B}} := \mathrm{udep}[2](x_3;\alpha_3) \vee (\alpha_3 = \underline{0} \wedge \mathrm{udep}[2](x_2;a_2)) \vee \mathrm{udep}[2](x_1,x_2,x_3;\alpha_1,\alpha_2,\alpha_3)$
- $\psi_{\mathfrak{B}} := (\alpha_1 = \underline{1} \wedge \alpha_2 = \underline{0} \wedge \alpha_3 = \underline{0}) \vee (\alpha_1 = \underline{0} \wedge \alpha_2 = \underline{1} \wedge \alpha_3 = \underline{0}) \vee (\alpha_1 = \underline{0} \wedge \alpha_2 = \underline{0} \wedge \alpha_3 = \underline{1})$
- $\nu_{\mathfrak{B}} := (\alpha_1 = \underline{1} \wedge \alpha_2 = \underline{1} \wedge \alpha_3 = \underline{1}) \vee (\alpha_1 = \underline{0} \wedge \alpha_2 = \underline{0} \wedge \alpha_3 = \underline{0}) \vee$
  $(\alpha_1 = \underline{1} \wedge \alpha_2 = \underline{1} \wedge \alpha_3 = \underline{0}) \vee (\alpha_1 = \underline{1} \wedge \alpha_2 = \underline{0} \wedge \alpha_3 = \underline{1}) \vee (\alpha_1 = \underline{0} \wedge \alpha_2 = \underline{1} \wedge \alpha_3 = \underline{1})$.

4 From Dependence Logic to MMSNP

In this section, we prove that MMSNP is at least as expressive as $\forall$-MUD$[\omega]$. In the proof of this result, we will make use of the extension of MMSNP with k-valued variables. This logic, denoted MMSNP$[k]$, is obtained from MMSNP by adding equalities of the form $\alpha = \underline{i}$, $i \in [k]$, and universal quantification of k-valued variables. As in the case of $\forall$-MUD$[k]$, there is no need to add equalities of the form $\alpha = \beta$, since they are definable.

We will use the logic MMSNP$[k]$ only as an intermediate step in translating formulas of $\forall$-MUD$[k]$ to equivalent sentences of MMSNP. The following lemma, which has a straightforward proof, tells that the k-valued variables in any MMSNP$[k]$-sentence can be eliminated to obtain an equivalent MMSNP-sentence.

▶ **Lemma 11.** *For every sentence $\varphi \in$ MMSNP$[k]$, there is a sentence $\psi \in$ MMSNP such that, for every structure $\mathfrak{A}$, we have that $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{A} \models \psi$.*

We make next the simple observation that uniform dependence atoms are expressible in existential monadic second-order logic. Indeed, the function $f : A \to [k]$ in the semantics of $\mathrm{udep}[k]$ can be replaced with a k-tuple of unary relations: $\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha})$ if and only if there are $P_1^{\mathfrak{A}}, \ldots, P_k^{\mathfrak{A}} \subseteq A$ such that

1. $A = \bigcup_{j\in[k]} P_j^{\mathfrak{A}}$ and $P_i^{\mathfrak{A}} \cap P_j^{\mathfrak{A}} = \emptyset$ for $i \neq j$, and
2. $s(x_i) \in P_{s(\alpha_i)}^{\mathfrak{A}}$ for all $i \in [n]$ and $s \in T$.

Both of these conditions can be expressed by first-order formulas:

1. $A = \bigcup_{j\in[k]} P_j^{\mathfrak{A}}$ and $P_i^{\mathfrak{A}} \cap P_j^{\mathfrak{A}} = \emptyset$ for $i \neq j$ if and only if $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k]}) \models \forall y\, \theta$, where

$$\theta := \bigvee_{j\in[k]} P_j(y) \wedge \bigwedge_{1\leq i<j\leq k} \neg(P_i(y) \wedge P_j(y)).$$

2. $s(x_i) \in P_{s(\alpha_i)}^{\mathfrak{A}}$ for all $i \in [n]$ and $s \in T$ if and only if $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k]}), s \models \rho_{k,n}$ for all $s \in T$, where

$$\rho_{k,n} := \bigwedge_{i\in[n],j\in[k]} (P_j(x_i) \leftrightarrow \alpha_i = \underline{j}).$$

Thus, $\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha})$ if and only if $\mathfrak{A}, s \models \exists P_1 \ldots \exists P_k(\forall y\, \theta \wedge \rho_{k,n})$ for all $s \in T$. In the next lemma, we formulate the definition of $\mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha})$ by a *sentence* of MMSNP$[k]$. To do this, we need to replace the team T with the corresponding relation $R_{T,\boldsymbol{x}\boldsymbol{\alpha}}$.

▶ **Lemma 12.** *Let $\mathfrak{A}$ be a structure and T a team on $\mathfrak{A}$ such that the domain of T is $V = \{x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_n\}$. Then the following statements are equivalent:*

1. $\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha})$.
2. $(\mathfrak{A}, R_{T,\boldsymbol{x}\boldsymbol{\alpha}}) \models \exists P_1 \ldots \exists P_k \forall \boldsymbol{x} \forall y \forall \boldsymbol{\alpha}(R(\boldsymbol{x}\boldsymbol{\alpha}) \to \theta \wedge \rho_{k,n})$.

Proof. Using the preceding observations, we get the following chain of equivalences:

$$\begin{aligned}
\mathfrak{A}, T \models \mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha}) \iff\ & \text{there are } P_1^{\mathfrak{A}}, \dots, P_k^{\mathfrak{A}} \subseteq A \text{ such that, for all } s \in T, \\
& (\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k]}), s \models \forall y\, \theta \wedge \rho_{k,n} \\
\iff\ & \text{there are } P_1^{\mathfrak{A}}, \dots, P_k^{\mathfrak{A}} \subseteq A \text{ such that, for all } s' \in F_{V\cup\{y\}}, \\
& (\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k]}, R_{T,\boldsymbol{x}\boldsymbol{\alpha}}), s' \models R(\boldsymbol{x}\boldsymbol{\alpha}) \to \theta \wedge \rho_{k,n} \\
\iff\ & (\mathfrak{A}, R_{T,\boldsymbol{x}\boldsymbol{\alpha}}) \models \exists P_1 \dots \exists P_k \forall \boldsymbol{x} \forall y \forall \boldsymbol{\alpha} (R(\boldsymbol{x}\boldsymbol{\alpha}) \to \theta \wedge \rho_{k,n}). \qquad \blacktriangleleft
\end{aligned}$$

Our next aim is to show that the translation of uniform dependence atoms to $\mathsf{MMSNP}[k]$ given in Lemma 12 can be extended to arbitrary formulas of $\mathsf{QF\text{-}MUD}[k]$. The number of existentially quantified unary relations in the translation of a formula ψ depends on the number of occurrences of dependence atoms in ψ. We denote this number by $\sharp_{\mathrm{dep}}(\psi)$.

▶ **Lemma 13.** *Let ψ be a formula of $\mathsf{QF\text{-}MUD}[k]$ with $\mathrm{Fr}(\psi) = \{x_1, \dots, x_n, \alpha_1, \dots, \alpha_m\}$ and $\sharp_{\mathrm{dep}}(\psi) = \ell$. There is a quantifier-free formula ψ^+ with $\mathrm{Fr}(\psi^+) = \mathrm{Fr}(\psi) \cup \{y_1, \dots, y_\ell\}$ and with free unary second-order variables $P_1, \dots, P_{k\ell}$ such that for every structure $\mathfrak{A}$ and every team T on $\mathfrak{A}$ with $\mathrm{Fr}(\psi) \subseteq \mathrm{dom}(T)$, we have that the following statements are equivalent:*
1. $\mathfrak{A}, T \models \psi$.
2. $(\mathfrak{A}, R_{T,\boldsymbol{x}\boldsymbol{\alpha}}) \models \exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x} \forall \boldsymbol{y} \forall \boldsymbol{\alpha} (R(\boldsymbol{x}\boldsymbol{\alpha}) \to \psi^+)$.

Proof. The claimed equivalence is proved by induction on the formula ψ.

- If ψ is a negated atomic formula $\neg S(\boldsymbol{x})$, then $m = \ell = 0$, and we have

$$\begin{aligned}
\mathfrak{A}, T \models \psi &\iff \mathfrak{A}, s \models \psi \text{ for every assignment } s \in T \\
&\iff (\mathfrak{A}, R_{T,\boldsymbol{x}}), s \models R(\boldsymbol{x}) \to \psi \text{ for every assignment } s \in F_{\mathrm{dom}(T)} \\
&\iff (\mathfrak{A}, R_{T,\boldsymbol{x}}), s \models \forall \boldsymbol{x}(R(\boldsymbol{x}) \to \psi).
\end{aligned}$$

  Thus, we simply let $\psi^+ := \psi$ in this case.
- If ψ is an equality $\alpha_1 = \underline{j}$, we define $\psi^+ := \psi$. Then $n = \ell = 0$, $m = 1$ and in the same way as in the previous case, we see that $\mathfrak{A}, T \models \psi \iff (\mathfrak{A}, R_{T,\alpha_1}), s \models \forall \alpha_1 (R(\alpha_1) \to \psi^+)$.
- If ψ is a uniform dependence atom $\mathrm{udep}[k](\boldsymbol{x};\boldsymbol{\alpha})$, we define $\psi^+ = \theta \wedge \rho_{k,n}$; the claimed equivalence follows from Lemma 12.
- Let $\psi = \pi \vee \sigma$. Then $\ell = \sharp_{\mathrm{dep}}(\psi) = \ell' + \ell''$, where $\ell' = \sharp_{\mathrm{dep}}(\pi)$ and $\ell'' = \sharp_{\mathrm{dep}}(\sigma)$. Let σ_*^+ be the formula obtained from σ^+ by replacing each relation variable P_j with $P_{k\ell'+j}$, $j \in [k\ell'']$, and replacing each first-order variable y_j with $y_{\ell'+j}$, $j \in [\ell'']$. Put $\psi^+ := \pi^+ \vee \sigma_*^+$.
  Assume first that $\mathfrak{A}, T \models \psi$. Then there are subteams T' and T'' of T such that $T = T' \cup T''$, $\mathfrak{A}, T' \models \pi$ and $\mathfrak{A}, T'' \models \sigma$. Let $\boldsymbol{u}$ and $\boldsymbol{\beta}$ be tuples that list those of the variables in $\{x_1, \dots, x_n, \alpha_1, \dots, \alpha_m\}$ that occur in π, and similarly let the tuples $\boldsymbol{v}$ and $\boldsymbol{\gamma}$ list the variables occurring in σ. By induction hypothesis, we have

$$(\mathfrak{A}, R_{T',\boldsymbol{u}\boldsymbol{\beta}}) \models \exists P_1 \dots \exists P_{k\ell'} \forall \boldsymbol{u} \forall \boldsymbol{y'} \forall \boldsymbol{\beta} (R(\boldsymbol{u}\boldsymbol{\beta}) \to \pi^+),$$

  where $\boldsymbol{y'} = (y_1, \dots, y_{\ell'})$. Similarly, changing the bound variables and using the induction hypothesis, we get $(\mathfrak{A}, R_{T'',\boldsymbol{v}\boldsymbol{\gamma}}) \models \exists P_{k\ell'+1} \dots \exists P_{k\ell} \forall \boldsymbol{v} \forall \boldsymbol{y''} \forall \boldsymbol{\gamma} (R(\boldsymbol{v}\boldsymbol{\gamma}) \to \sigma_*^+)$, where $\boldsymbol{y''} = (y_{\ell'+1}, \dots, y_\ell)$. Thus, there are relations $P_1^{\mathfrak{A}}, \dots, P_{k\ell}^{\mathfrak{A}} \subseteq A$ such that
  - $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \pi^+$ for every assignment $s \in T'[A^\ell/\boldsymbol{y}]$, and
  - $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \sigma_*^+$ for every assignment $s \in T''[A^\ell/\boldsymbol{y}]$.

  Since $T = T' \cup T''$, it follows that $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \pi^+ \vee \sigma_*^+$ for all assignments $s \in T[A^\ell/\boldsymbol{y}]$. Thus, we see that $(\mathfrak{A}, R_{T,\boldsymbol{x}\boldsymbol{\alpha}}) \models \exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x} \forall \boldsymbol{y} \forall \boldsymbol{\alpha} (R(\boldsymbol{x}\boldsymbol{\alpha}) \to (\pi^+ \vee \sigma_*^+))$.

For the other direction, assume that there are relations $P_1^{\mathfrak{A}}, \dots, P_{k\ell}^{\mathfrak{A}} \subseteq A$ such that

$$(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}, R_{T,\boldsymbol{x\alpha}}) \models \forall \boldsymbol{x}\forall \boldsymbol{y}\forall \boldsymbol{\alpha}(R(\boldsymbol{x\alpha}) \to (\pi^+ \vee \sigma_*^+)).$$

Then $(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \pi^+ \vee \sigma_*^+$ for every assignment $s \in T[A^\ell/\boldsymbol{y}]$. Let T' and T'' be the subteams

- $T' := \{s \in T \mid (\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \forall \boldsymbol{y}'\pi^+\}$ and
- $T'' := \{s \in T \mid (\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell]}), s \models \forall \boldsymbol{y}''\sigma_*^+\}$,

where $\boldsymbol{y}'$ and $\boldsymbol{y}''$ are as above. By the definition of T', we have

$$(\mathfrak{A}, (P_j^{\mathfrak{A}})_{j\in[k\ell']}, R_{T',\boldsymbol{u\beta}}) \models \forall \boldsymbol{u}\forall \boldsymbol{y}'\forall \boldsymbol{\beta}(R(\boldsymbol{u\beta}) \to \pi^+).$$

Universal quantification can be restricted here to those first-order and k-valued variables that occur in π^+. Similarly, it suffices to consider relation variables P_j occurring in π^+. From the induction hypothesis, it follows that $\mathfrak{A}, T' \models \pi$. In the same way, we see that

$$(\mathfrak{A}, (P_{k\ell'+j}^{\mathfrak{A}})_{j\in[k\ell'']}, R_{T'',\boldsymbol{v\gamma}}) \models \forall \boldsymbol{v}\forall \boldsymbol{y}''\forall \boldsymbol{\gamma}(R(\boldsymbol{v\gamma}) \to \sigma_*^+).$$

Replacing each relation variable $P_{k\ell'+j}$ by P_j and each first-order variable $y_{\ell'+j}$ by y_j and using the induction hypothesis, we see that $\mathfrak{A}, T'' \models \sigma$. Finally, it is clear that $T = T' \cup T''$, whence we get $\mathfrak{A}, T \models \psi$.

- If $\psi = \pi \wedge \sigma$, we define $\psi^+ := \pi^+ \wedge \sigma_*^+$. This case is handled as the preceding one. ◀

We now have all the technical machinery needed to prove the main result of this section.

▶ **Theorem 14** (∀-MUD[ω] is contained in MMSNP). *For every sentence $\varphi \in \forall$-MUD[ω] there is a sentence $\varphi^* \in$ MMSNP such that the following statements are equivalent:*

1. $\mathfrak{A} \models \varphi$.
2. $\mathfrak{A} \models \varphi^*$.

Proof. By Lemma 11, it suffices to prove the claimed equivalence for a sentence φ^* of MMSNP[k]. Let ψ be a formula of QF-MUD[k] such that φ is (equivalent to) its universal closure $\forall x_1 \dots \forall x_n \forall \alpha_1 \dots \forall \alpha_m \psi$, and let F be the full team on $\mathfrak{A}$ with domain $\{x_1, \dots, x_n, \alpha_1, \dots, \alpha_m\}$. By applying Lemma 13, we obtain a quantifier-free formula ψ^+ such that

$$\mathfrak{A}, F \models \psi \iff (\mathfrak{A}, R_{F,\boldsymbol{x\alpha}}) \models \exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x}\forall \boldsymbol{y}\forall \boldsymbol{\alpha}(R(\boldsymbol{x\alpha}) \to \psi^+).$$

Since F is the full team, $(\mathfrak{A}, R_{F,\boldsymbol{x\alpha}}), s \models R(\boldsymbol{x\alpha})$ holds for all assignments s. Hence, the implication $R(\boldsymbol{x\alpha}) \to \psi^+$ can be replaced by its right-hand side ψ^+, and the relation $R_{F,\boldsymbol{x\alpha}}$ can be omitted in the equivalence above. Thus, using Lemma 6, we obtain the equivalence

$$\mathfrak{A} \models \varphi \iff \mathfrak{A} \models \exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x}\forall \boldsymbol{y}\forall \boldsymbol{\alpha}\psi^+.$$

From the definition of ψ^+, we see that the sentence $\exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x}\forall \boldsymbol{y}\forall \boldsymbol{\alpha}\, \psi^+$ is in MMSNP[k]. Thus, the claimed equivalence holds for $\varphi^* := \exists P_1 \dots \exists P_{k\ell} \forall \boldsymbol{x}\forall \boldsymbol{y}\forall \boldsymbol{\alpha}\, \psi^+$. ◀

We note that the size of the MMSNP[k]-sentence φ^* is polynomial in the size of the ∀-MUD[ω]-sentence φ. In fact, it is not hard to show that the size of φ^* is $O(nk^3|\varphi|)$.

Given a sentence $\varphi \in \forall$-MUD[ω], we denote its model-checking problem "given a structure $\mathfrak{A}$, does $\mathfrak{A} \models \varphi$ hold?" by $\mathcal{MC}(\varphi)$. Using the polynomial-time equivalence of MMSNP and CSP, we obtain the following corollary to Theorems 10 and 14.

▶ **Corollary 15.** *If φ is a sentence of ∀-MUD[ω], then $\mathcal{MC}(\varphi)$ is polynomial-time equivalent to CSP($\mathfrak{B}$), for some structure $\mathfrak{B}$; vice versa, if $\mathfrak{B}$ is a structure, then CSP($\mathfrak{B}$) is polynomial-time equivalent to $\mathcal{MC}(\varphi)$, for some sentence φ of ∀-MUD[ω].*

5 Concluding Remarks

In this paper, we established a tight connection between dependence logic and constraint satisfaction. Since dependence logic has the same expressive power as existential second-order logic, it is expected that constraint satisfaction problems can be expressed in dependence logic. We believe, however, that the connection established in this paper is a priori unexpected, since we showed that a simple fragment of *universal* dependence logic captures, in a precise sense, the family of constraint satisfaction problems CSP($\mathfrak{B}$), where $\mathfrak{B}$ is a relational structure. Our results contribute to the descriptive complexity of constraint satisfaction and also shed new light on quantifier-free and universal dependence logic.

The connection between universal dependence logic and constraint satisfaction is established by using MMSNP as a bridge and also the result by Feder and Vardi [5] that MMSNP captures constraint satisfaction via polynomial-time reductions. Specifically, we showed that every constraint satisfaction problem CSP($\mathfrak{B}$), in which $\mathfrak{B}$ has only one relation, is definable by a $\forall$-MUD[ω]-sentence, and every $\forall$-MUD[ω]-sentence is equivalent to some MMSNP-sentence. A natural question that arises from these results is whether every MMSNP-sentence is equivalent to some $\forall$-MUD[ω]-sentence or, in other words, whether MMSNP and $\forall$-MUD[ω] have the same expressive power. A related question is to identify other natural fragments of dependence logic that capture important fragments of existential second-order logic, such as *strict* existential second-order logic (i.e., the fragment of existential second-order logic in which all first-order quantifiers are universal).

Acknowledgements. A part of the research reported here was carried out while Lauri Hella was visiting the University of California Santa Cruz.

References

1 Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors. *Complexity of Constraints – An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*. Springer, 2008.

2 Arnaud Durand, Juha Kontinen, Nicolas de Rugy-Altherre, and Jouko Väänänen. Tractability frontier of data complexity in team semantics. In *Proceedings Sixth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2015, Genoa, Italy, 21-22nd September 2015.*, pages 73–85, 2015.

3 Paul Erdös. Graph theory and probability. *Canadian J. of Mathematics*, 11:34–38, 1959.

4 Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard Karp, editor, *Complexity of Computation*, number 7 in SIAM-AMS Proceedings, pages 43–73. SIAM-AMS, 1974.

5 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

6 P. Galliani. Inclusion and exclusion dependencies in team semantics – on some logics of imperfect information. *Ann. Pure Appl. Logic*, 163(1):68–84, 2012.

7 Pietro Galliani and Lauri Hella. Inclusion logic and fixed point logic. In *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, number 23 in LIPIcs, pages 281–295. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.281.

8 Erich Grädel and Jouko A. Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013. doi:10.1007/s11225-013-9479-2.

9 Johan Håstad, Andrei A. Krokhin, and Dániel Marx. The constraint satisfaction problem: Complexity and approximability (Dagstuhl Seminar 12451). *Dagstuhl Reports*, 2(11):1–19, 2012.

10 Leon Henkin. Some remarks on infinitely long formulas. In *Infinitistic Methods*. Pergamon Press, 1961.

11 Jaakko Hintikka and Gabriel Sandu. Informational independence as a semantical phenomenon. In J. E. Fenstad et al., editor, *Logic, Methodology and the Philosophy of Science VIII*, pages 571–89. North-Holland, 1989.

12 Jarmo Kontinen. Coherence and computational complexity of quantifier-free dependence logic formulas. *Studia Logica*, 101(2):267–291, 2013. doi:10.1007/s11225-013-9481-8.

13 Juha Kontinen and Jouko A. Väänänen. On definability in dependence logic. *Journal of Logic, Language and Information*, 18(3):317–332, 2009. doi:10.1007/s10849-009-9082-0.

14 Juha Kontinen and Jouko A. Väänänen. Axiomatizing first-order consequences in dependence logic. *Ann. Pure Appl. Logic*, 164(11):1101–1117, 2013. doi:10.1016/j.apal.2013.05.006.

15 Gábor Kun and Jaroslav Nesetril. Forbidden lifts (NP and CSP for combinatorialists). *Eur. J. Comb.*, 29(4):930–945, 2008.

16 Richard E. Ladner. On the structure of polynomial time reducibility. *J. ACM*, 22(1):155–171, 1975.

17 Jouko A. Väänänen. *Dependence Logic – A New Approach to Independence Friendly Logic*, volume 70 of *London Mathematical Society student texts*. Cambridge University Press, 2007. URL: http://www.cambridge.org/de/knowledge/isbn/item1164246/.

Quantified Constraint Satisfaction on Monoids*

Hubie Chen[1] and Peter Mayr[2]

1 **University of the Basque Country (UPV/EHU), E-20018 San Sebastián, Spain; and**
IKERBASQUE, Basque Foundation for Science, E-48011 Bilbao, Spain
hubie.chen@ehu.es

2 **Department of Mathematics, University of Colorado Boulder, Campus Box 395, Boulder, CO 80309-0395, USA**
peter.mayr@colorado.edu

Abstract

We contribute to a research program that aims to classify, for each finite structure, the computational complexity of the quantified constraint satisfaction problem on the structure. Employing an established algebraic viewpoint to studying this problem family, whereby this classification program can be phrased as a classification of algebras, we give a complete classification of all finite monoids.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases quantified constraint satisfaction, universal algebra, computational complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.15

1 Introduction

Problem frameworks. The *constraint satisfaction problem (CSP)* is a generic combinatorial problem where an input consists of a set of constraints on a set of variables, and the question is to determine whether or not there is an assignment to the variables satisfying all of the constraints. The CSP can be formulated logically as the problem of deciding, given an *existential conjunctive sentence* and a finite structure, whether or not the sentence evaluates to true on the structure. By an *existential conjunctive sentence*, we mean a first-order sentence built from atoms, conjunction ($\wedge$), and existential quantification ($\exists$).

It is well-known that the CSP is NP-complete in general. However, one can define, for each finite structure $\mathbb{B}$, the problem CSP($\mathbb{B}$) to be the restricted version of the CSP where the structure is fixed as $\mathbb{B}$, and in this way obtain a rich family of problems, some of which are polynomial-time decidable and hence escape the general intractability of the CSP. This family includes a variety of well-established combinatorial problems, such as graph homomorphism problems (for examples, *2-colorability*, *3-colorability*, and various generalizations thereof); Boolean satisfiability problems (for examples, *2-SAT* and *Horn-SAT*); and algebraic equations problems.

A natural generalization of the CSP is the *quantified constraint satisfaction problem (QCSP)*, where the task is to decide whether or not a *quantified conjunctive sentence* holds true on a structure. By a *quantified conjunctive sentence*, we mean a first-order sentence built

* The first author was supported by the Spanish Project MINECO COMMAS TIN2013-46181-C2-R, Basque Project GIU15/30, and Basque Grant UFI11/45; the second by the Austrian Science Fund (FWF): P24285.

© Hubie Chen and Peter Mayr;
licensed under Creative Commons License CC-BY

from atoms, conjunction ($\wedge$), and both quantifiers ($\exists, \forall$); so, the QCSP may be described as the generalization of the CSP where both quantifiers are permitted, as opposed to just existential quantifiers. It is well-known that the QCSP is PSPACE-complete in general. For each finite structure $\mathbb{B}$, one may also define QCSP($\mathbb{B}$) to be the restricted version of the QCSP where the structure is fixed as $\mathbb{B}$. The resulting problem family includes quantified generalizations of the mentioned combinatorial problems. (For a broader perspective on model checking various fragments of first-order logic on fixed finite structures, we refer the reader to [18].)

Classification. The problem families CSP($\mathbb{B}$) and QCSP($\mathbb{B}$) give rise to fundamental classification programs, in which researchers aim to describe the complexity of each of the problems in the respective families. Perhaps the best-known result in this vein is Schaefer's dichotomy theorem [22], which shows for each two-element structure $\mathbb{B}$, the problem CSP($\mathbb{B}$) is polynomial-time decidable if the structure $\mathbb{B}$ satisfies one of six presented conditions, and is NP-complete otherwise. Schaefer also claimed without proof a partial classification of the problems QCSP($\mathbb{B}$), over all two-element structures $\mathbb{B}$; a complete classification was presented and proved in the book [12]. Current research on the CSP [16, 1] and the QCSP [7, 19, 8, 9, 17, 6] aims to understand the behavior of all finite structures with respect to each of these two problem families, without any universe size restriction of the type employed by Schaefer. At the present juncture, it seems fair to suggest that the family QCSP($\mathbb{B}$) is less understood than the family CSP($\mathbb{B}$). For examples of evidence, we observe that the classification of CSP($\mathbb{B}$) over all undirected graphs is a classical result due to Hell and Nesetril [14], but no such classification is known for QCSP($\mathbb{B}$) (although partial results exist [20, 9]); also, the classification of CSP($\mathbb{B}$) over all three-element structures was given by Bulatov [5], but no such classification is known for QCSP($\mathbb{B}$) (see [7, 8] for partial results).

A significant stimulus for these classification programs was the introduction of an algebraic approach [3] that permits the deployment of notions, concepts, and results from universal algebra. A cornerstone of this algebraic approach is the passage from a structure $\mathbb{B}$ to an algebra, the so-called algebra of *polymorphisms* of $\mathbb{B}$. It is known that, intuitively speaking, this algebra retains the relevant information about the complexity of the structure, in the precise sense that two structures $\mathbb{B}, \mathbb{B}'$ sharing the same algebra enjoy that the problems CSP($\mathbb{B}$) and CSP($\mathbb{B}'$) are interreducible [3], and similarly that the problems QCSP($\mathbb{B}$) and QCSP($\mathbb{B}'$) are interreducible [2]. (Here, we mean *interreducible* with respect to many-one polynomial-time reduction). In fact, each of the described classification programs on structures can be rephrased as a classification program on algebras (see [3]).

An initial basic result of the algebraic approach to the CSP states that, for each structure $\mathbb{B}$, there exists a structure $\mathbb{B}'$ whose algebra is *idempotent*, such that CSP($\mathbb{B}$) and CSP($\mathbb{B}'$) are interreducible [3]. An algebra is *idempotent* if each operation f thereof is idempotent in that $f(a, \ldots, a) = a$ holds for each element a. On the algebraic side, this result implies that one can restrict to studying and to classifying idempotent algebras in order to carry out the classification program on the problem family CSP($\mathbb{B}$). In contrast to this state of affairs for the CSP, the QCSP has not seen any such result that allows attention to be restricted to idempotent algebras, although attempts have been made to understand this discrepancy [11]. Despite the lack of such a result for the QCSP, we believe it fair to claim that the development of the algebraic approach for the QCSP has focused on idempotent algebras [7, 8, 9, 6]. Non-idempotent algebras hence constitute a *terra incognita* in QCSP research.

Contribution. The present article was motivated by the desire to initiate a systematic study of the *terra incognita* of non-idempotent algebras, with respect to QCSP complexity. In this article, we investigate the class of finite monoids. Recall that a *semigroup* is an algebra comprised of a set equipped with an associative binary operation; a *monoid* is a semigroup whose operation has an identity element.

Our main theorem is the complete classification of finite monoids with respect to QCSP complexity. We identify a simply stated algebraic condition on monoids, and show that each monoid satisfying this condition has a polynomial-time tractable QCSP; we show that all other monoids have an NP-complete QCSP. (See Theorem 7 for a precise statement.) We remark that the fact that each monoid has a QCSP in NP is due to previous work (see Theorem 6). We elected to focus on classifying monoids since their CSP complexity was already understood—indeed, semigroups were historically one of the first classes of algebras to be understood for the CSP [4]—and also because this would permit the usage of the established structure theory of semigroups. Indeed, our work opens up an interface between quantified constraint satisfaction and semigroup theory that, we believe, presents new perspectives on each of these two fields. One intriguing aspect of our dichotomy theorem is this: the algebraic condition that we identify concerns whether or not the monoid is generated by a particular subset of its elements. At the same time, the size of generating sets for algebras was previously linked to QCSP complexity, in particular, to the study of which problems QCSP($\mathbb{B}$) are in NP [8, 9].

2 Preliminaries

2.1 General preliminaries

When $\mathcal{A}, \mathcal{B}$ are decision problems such that there is a polynomial-time many-one reduction from $\mathcal{B}$ to $\mathcal{A}$, we write $\mathcal{B} \leq_p^m \mathcal{A}$. When I is an instance of a decision problem and I' is also an instance of a decision problem, we say that I and I' are *decision-equivalent* when I is a *yes* instance iff I' is a *yes* instance.

2.2 Semigroups

We recall some notation and basic facts for semigroups that will be used in this paper (see [15]). A *semigroup* $\mathbf{S} = \langle S, \cdot \rangle$ is a non-empty set S with an associative binary operation $\cdot$, the multiplication. A *subsemigroup* of $\mathbf{S}$ is a non-empty subset $T \subseteq S$ that is closed under multiplication. An element $x \in S$ is *idempotent* if $x^2 = x$. We will use the fact that, when $\mathbf{S}$ is a finite monoid of size n, for each element $x \in S$, it holds that $x^{n!}$ is idempotent. If $\mathbf{S}$ contains an element 1 such that $x \cdot 1 = 1 \cdot x = x$ for all $x \in S$, we call 1 the *identity* of $\mathbf{S}$ and $\mathbf{S}$ a *monoid*. If $\mathbf{S}$ has no identity, we can adjoin an extra element 1 to S to form a monoid $\mathbf{S}^1$. For that, let S^1 denote $S \cup \{1\}$ and extend the multiplication of $\mathbf{S}$ by $1 \cdot x = x \cdot 1 = x$ for all $x \in S^1$. If $\mathbf{S}$ already contains an identity, let $\mathbf{S}^1 = \mathbf{S}$. An element $a \in S$ is *regular* if there exists $x \in S$ such that $axa = a$. A semigroup is *zero* if its multiplication is constant.

Let $\mathbf{S}$ be a semigroup. An *ideal* is a non-empty subset $I \subseteq S$ such that $SI := \{si \ : \ s \in S, i \in I\} \subseteq I$ and $IS := \{is \ : \ s \in S, i \in I\} \subseteq I$. The ideal generated by an element $a \in S$ is hence S^1aS^1. The equivalence relation $\mathcal{J}$ on S is defined by $a \mathcal{J} b$ iff $S^1aS^1 = S^1bS^1$. The equivalence classes of the equivalence relation $\mathcal{J}$ are called $\mathcal{J}$*-classes*, and the $\mathcal{J}$-class containing the element a is denoted by J_a. There is a natural partial order on $\mathcal{J}$-classes, given by $J_a \leq J_b$ iff $S^1aS^1 \subseteq S^1bS^1$.

A *right ideal* of $\mathbf{S}$ is a non-empty subset $A \subseteq S$ such that $AS \subseteq A$. The right ideal generated by an element $a \in S$ is aS^1. The equivalence relation $\mathcal{R}$ on S is defined by $a\mathcal{R}b$ iff $aS^1 = bS^1$. In an analogous fashion, we define *left ideals* and the equivalence $\mathcal{L}$.

Given an ideal I of a semigroup $\mathbf{S} = \langle S, \cdot \rangle$, the *Rees quotient* $\mathbf{S}/I$ is the semigroup on $(S \setminus I) \cup \{0\}$ with multiplication

$$xy := \begin{cases} x \cdot y & \text{if } x, y \in S \text{ and } x \cdot y \in S \setminus I, \\ 0 & \text{otherwise.} \end{cases}$$

For a group $\mathbf{G}$, sets I, Λ and a $\Lambda \times I$-matrix P with entries in $G \cup \{0\}$ the *Rees matrix semigroup* $\mathbf{M} := M^0(\mathbf{G}, I, \Lambda, P)$ is defined on the set

$$M^0 = \{(i, g, \lambda) \ : \ i \in I, g \in G, \lambda \in \Lambda\} \cup \{0\}$$

with multiplication

$$(i, g, \lambda)(j, h, \mu) := \begin{cases} (i, gP_{\lambda,j}h, \mu) & \text{if } P_{\lambda,j} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

for all $(i, g, \lambda), (j, h, \mu) \in M^0$ and by $x0 = 0x = 0$ for all $x \in M^0$. The *Rees matrix semigroup* $\mathbf{M} := M(\mathbf{G}, I, \Lambda, P)$ is defined similarly on $M = \{(i, g, \lambda) \ : \ i \in I, g \in G, \lambda \in \Lambda\}$ when P has entries in G.

We collect some well-known facts on $\mathcal{J}$-classes of finite semigroups from Section 3 of [15] in the following lemma.

▶ **Lemma 1.** *Let $\mathbf{S}$ be a finite semigroup and $a \in S$.*

1. *If a is not regular, then J_a is not minimal and $(S^1aS^1)/\bigcup\{J_x \ : \ J_x < J_a\}$ is a zero semigroup.*
2. *If a is regular and J_a not minimal, then $(S^1aS^1)/\bigcup\{J_x \ : \ J_x < J_a\}$ is isomorphic to a Rees matrix semigroup $M^0(\mathbf{G}, I, \Lambda, P)$ for some group $\mathbf{G}$, index sets I, Λ and a $\Lambda \times I$-matrix P with entries in $G \cup \{0\}$ such that every row and every column contains an element from G.*
3. *If J_a is the minimal $\mathcal{J}$-class, then it is isomorphic to a Rees matrix semigroup $M(\mathbf{G}, I, \Lambda, P)$ for some group $\mathbf{G}$, index sets I, Λ and a $\Lambda \times I$-matrix P with entries in G.*

Let $\mathbf{S}$ be a semigroup. Then $\mathbf{S}$ is a *block group* if for all idempotents $e, f \in S$:

$$ef = e, fe = f \Rightarrow e = f,$$
$$ef = f, fe = e \Rightarrow e = f.$$

Equivalently, $\mathbf{S}$ is a block group if every $\mathcal{L}$-class and every $\mathcal{R}$-class of $\mathbf{S}$ contains at most one idempotent. We will make use of the following facts on block groups.

▶ **Lemma 2.** *Let $\mathbf{S}$ be a finite block group with $a \in S$ regular.*

1. *If J_a is not minimal, then $(S^1aS^1)/\bigcup\{J_x \ : \ J_x < J_a\}$ is isomorphic to a Rees matrix semigroup $M^0(\mathbf{G}, I, I, P)$ for some group $\mathbf{G}$, an index set I and the identity matrix P.*
2. *If J_a is the smallest $\mathcal{J}$-class, then it forms a group $\mathbf{G}$.*
3. *There exist unique idempotents $e \in R_a$, $f \in L_a$ such that $ea = af = a$.*

Proof. Item (1) follows from Lemma 1 (2): $(S^1aS^1)/\bigcup\{J_x \ : \ J_x < J_a\}$ is some $M^0(\mathbf{G}, I, \Lambda, P)$. Note that whenever $P_{\lambda,i} \neq 0$ for some $i \in I, \lambda \in \Lambda$, then $e = (i, P^{-1}_{\lambda,i}, \lambda)$ is idempotent. Suppose $f = (j, P^{-1}_{\lambda,j}, \lambda)$ for $j \in I$ is idempotent as well. Then $ef = e$ and $fe = f$. Since $\mathbf{S}$ is

a block group this implies $e = f$ and $i = j$. Consequently $P_{\lambda,j} = 0$ for all $j \neq i$. Hence every row of P (and similarly every column) contains at most one non-zero entry. Together with Lemma 1 (2) it follows that every row and every column of P contains exactly one entry from G, the remaining being 0. Hence $|I| = |\Lambda|$. By reordering rows and columns and normalizing P, we obtain that $(S^1aS^1)/\bigcup\{J_x \ : \ J_x < J_a\}$ is isomorphic to $M^0(\mathbf{G}, I, I, P)$ with P the identity matrix.

By a similar argument Lemma 1 (3) implies item (2). We have that J_a is isomorphic to $M(\mathbf{G}, I, \Lambda, P)$ for some group $\mathbf{G}$, index sets I, Λ and a $\Lambda \times I$-matrix P with entries in G. Since $\mathbf{S}$ is a block group, P contains at most one entry in every row and column. So $|I| = |\Lambda| = 1$ and $M(\mathbf{G}, I, \Lambda, P)$ is isomorphic to $\mathbf{G}$.

For (3) we may identify the elements of J_a with $\{(i, g, j) \ : \ i, j \in I, g \in G\}$ by (1), (2), respectively. Let $a = (i, g, j)$ for some $i, j \in I, g \in G$. Then $e := (i, 1, i)$, $f := (j, 1, j)$ are the unique idempotents in R_a, L_a, respectively. Clearly $ea = af = a$. ◀

2.3 Algebra and constraint satisfaction

An *algebra* $\mathbf{A} = \langle A, F\rangle$ is a non-empty set A paired with a set of finitary operations F on A. A subset $R \subseteq A^n$ is an n-ary *relation* on A. A k-ary operation $f\colon A^k \to A$ *preserves* or *is a polymorphism of* an n-ary relation R if for any choice of k tuples $(a_1^1, \ldots, a_n^1), \ldots, (a_1^k, \ldots, a_n^k) \in R$, it holds that the tuple $(f(a_1^1, \ldots, a_1^k), \ldots, f(a_n^1, \ldots, a_n^k))$ is in R. We say that a relation is *preserved* by an algebra if it is preserved by all operations of the algebra, and we say that a relational structure is *preserved* by an algebra if each of its relations is preserved by the algebra. (We remark that, algebraically, a non-empty relation is preserved by an algebra iff it is the universe of a subalgebra of a power of the algebra.) Here, a *relational signature* is a set of *relation symbols*, each having an associated arity, and a *relational structure* $\mathbb{B}$ over a relational signature σ provides a non-empty set B and an interpretation $R^{\mathbb{B}} \subseteq B^k$ for each relation symbol $R \in \sigma$; here, k denotes the arity of R. We assume that each relational signature is finite, and that each relation of a relational structure is represented using a list of its tuples. If an algebra $\mathbf{A}$ preserves a relational structure $\mathbb{B}$, then the universe of $\mathbb{B}$ is a subset of the universe of $\mathbf{A}$. For a more detailed study on which relational structures are preserved by semigroups we refer to [21].

In this article, we deal with relational first-order logic. Define a *qcsp-sentence* to be a first-order sentence of the form $Q_1 v_1 \ldots Q_n v_n \phi$ where each Q_i is a quantifier in $\{\forall, \exists\}$; the v_i are variables, assumed to be pairwise distinct; and ϕ is a conjunction of atoms. By an *atom*, we refer to the application $R(w_1, \ldots, w_k)$ of a relation symbol to a tuple of variables. Define a *csp-sentence* to be a qcsp-sentence in which all quantifiers are existential.

When $\mathbf{A}$ is an algebra, we define the problem QCSP($\mathbf{A}$) as follows. An instance of QCSP($\mathbf{A}$) is pair $(\Phi, \mathbb{B})$ where $\mathbb{B}$ is a relational structure preserved by $\mathbf{A}$ and Φ is a qcsp-sentence over the signature of $\mathbb{B}$. The question is to decide whether or not $\mathbb{B} \models \Phi$. When $\mathbf{A}$ is an algebra, the problem CSP($\mathbf{A}$) is defined to be the restriction of QCSP($\mathbf{A}$) to instances $(\Phi, \mathbb{B})$ where Φ is a csp-sentence. Let $\mathbf{A}$ be an algebra, and let $(\exists x_1 \ldots \exists x_n \phi, \mathbb{B})$ be an instance of CSP($\mathbf{A}$); a mapping $f : \{x_1, \ldots, x_n\} \to B$ such that $\mathbb{B}, f \models \phi$ is called a *solution* of the instance. In the case that $\mathbf{A}$ is a semigroup, a solution is called *idempotent* if each element in its image is an idempotent of $\mathbf{A}$. A fact that we will tacitly use is the following.

▶ **Proposition 3.** *When* $\mathbf{S}$ *is a semigroup and* f *and* g *are solutions to an instance of* CSP($\mathbf{S}$)*, the assignment* fg *obtained by point-wise product is also a solution to the instance.*

We define QCSP($\mathbb{B}$) to be the problem of deciding, given a qcsp-sentence Φ over the signature of $\mathbb{B}$, whether or not $\mathbb{B} \models \Phi$; we define CSP($\mathbb{B}$) analogously, but with respect to csp-sentences.

The following is the statement of the classification of finite semigroups with respect to the CSP.

▶ **Theorem 4** ([4]). *Let* $\mathbf{S}$ *be a finite semigroup. If* $\mathbf{S}$ *is a block group, then* CSP($\mathbf{S}$) *is polynomial-time decidable; otherwise,* CSP($\mathbf{S}$) *is* NP*-complete.*

Let us now identify some results on quantified constraint satisfaction of which we will make use.

Let $\mathbf{A} = \langle A, F\rangle$ be an algebra. A *congruence* of $\mathbf{A}$ is an equivalence relation $\theta \subseteq A \times A$ that is preserved by $\mathbf{A}$. Suppose that θ is a congruence of $\mathbf{A}$. Denote the equivalence class of θ containing $a \in A$ by a^θ; then, for each $f \in F$, the operation f^θ given by $f^\theta(a_1^\theta, \ldots, a_k^\theta) = (f(a_1, \ldots, a_k))^\theta$ is well-defined. Define A^θ as $\{a^\theta \mid a \in A\}$ and F^θ as $\{f^\theta \mid f \in F\}$. A *homomorphic image* of $\mathbf{A}$ is an algebra of the form $\langle A^\theta, F^\theta\rangle$, where θ is a congruence of $\mathbf{A}$. The following result seems to have a folklore status in the area, but we are not aware of any proof in the literature, so we provide one here.

▶ **Lemma 5.** *Let* $\mathbf{A}$ *be an algebra, and let* $\mathbf{B}$ *be a homomorphic image of* $\mathbf{A}$*. Then* $\mathrm{QCSP}(\mathbf{B}) \leq_p^m \mathrm{QCSP}(\mathbf{A})$*.*

Proof. Let θ be a congruence of $\mathbf{A}$ such that $\mathbf{B}$ has the form $\langle A^\theta, F^\theta\rangle$. Let h be the mapping from A to B defined by $h(a) = a^\theta$. Let $(\Psi, \mathbb{B})$ be an instance of QCSP($\mathbf{B}$). For each relation $R^{\mathbb{B}}$ of $\mathbb{B}$, let k denote its arity, and define $R^{\mathbb{A}}$ as the relation $\{(a_1, \ldots, a_k) \in A^k \mid (h(a_1), \ldots, h(a_k)) \in R^{\mathbb{B}}\}$. Since the relations of $\mathbb{A}$ are preimages under the homomorphism h of the relations of $\mathbb{B}$, we have that $\mathbb{A}$ is preserved by $\mathbf{A}$. So the reduction outputs the instance $(\Psi, \mathbb{A})$ of QCSP($\mathbf{A}$).

We argue the correctness of this reduction as follows. Let Φ be a formula having the form $Q_1 v_1 \ldots Q_m v_m \phi$, where ϕ is a conjunction of atoms. It is straightforward to verify the following by induction on m: for any assignment g mapping variables to A, it holds that $\mathbb{A}, g \models \Phi$ iff $\mathbb{B}, h(g) \models \Phi$. Here, $h(g)$ denotes the composition of h and g. It follows that, when Φ is a qcsp-sentence, $\mathbb{A} \models \Phi$ iff $\mathbb{B} \models \Phi$. ◀

When $\mathbf{A}$ is an algebra, define $\mathrm{QCSP}_\forall(\mathbf{A})$ to be the restriction of QCSP($\mathbf{A}$) to instances $(\Phi, \mathbb{B})$ where Φ has at most one universally quantified variable.

▶ **Theorem 6** (follows from [7]). *Let* $\mathbf{S}$ *be a finite monoid. There exists a polynomial-time algorithm that, given as input an instance* $(\Phi, \mathbb{B})$ *of the problem* QCSP($\mathbf{S}$)*, outputs a set* $\mathcal{I}$ *of* $\mathrm{QCSP}_\forall(\mathbf{S})$ *instances such that* $(\Phi, \mathbb{B})$ *is a* yes *instance of* QCSP($\mathbf{S}$) *if and only if every instance in* $\mathcal{I}$ *is a* yes *instance of* $\mathrm{QCSP}_\forall(\mathbf{S})$*. Consequently, the problem* QCSP($\mathbf{S}$) *is in* NP*.*

This theorem can be established using Theorem 4.3 of [7]; let us explain how. Let e denote the identity element of the monoid $\mathbf{S}$; by [7, Theorem 4.3] each structure $\mathbb{B}$ that is preserved by a monoid $\mathbf{S}$ of size 2 is (in the language of [7]) $(1, e)$-collapsible. The same proof works for monoids of arbitrary sizes. It follows that deciding an instance of QCSP($\mathbf{S}$) is equivalent to checking if all of its $(1, e)$-collapsings are yes instances (see Definitions 3.1 and 3.11 of [7]), and these can be formulated as instances of $\mathrm{QCSP}_\forall(\mathbf{S})$. Note that the relation $\{e\}$ is preserved by $\mathbf{S}$, and so can be used in an atom to force a variable to take on the value e. Finally we have polynomially many instances of $\mathrm{QCSP}_\forall(\mathbf{S})$, each of which clearly are in NP. Hence QCSP($\mathbf{S}$) is in NP.

3 Dichotomy theorem statement

The following is the main classification result of this paper.

▶ **Theorem 7.** *Let* $\mathbf{S}$ *be a finite monoid.*

- *If* $\mathbf{S}$ *is a block group and generated by its regular elements, then the problem* QCSP($\mathbf{S}$) *is polynomial-time decidable.*
- *Otherwise, the problem* QCSP($\mathbf{S}$) *is* NP*-complete.*

We give a proof that makes forward references to the main theorems of the next two sections, Theorems 8 and 9.

Proof. By Theorem 6, the problem QCSP($\mathbf{S}$) is in NP. When $\mathbf{S}$ is a block group and generated by its regular elements, it follows from Theorem 9 that QCSP($\mathbf{S}$) is polynomial-time decidable. If $\mathbf{S}$ is not a block group, then QCSP($\mathbf{S}$) is NP-hard by Theorem 4; if $\mathbf{S}$ is not generated by its regular elements, then QCSP($\mathbf{S}$) is NP-hard by Theorem 8. ◀

We now describe how some concrete classes of monoids behave with respect to our dichotomy, but first, we need some more definitions. A semigroup is *inverse* if all its elements are regular and its idempotents commute. Hence inverse semigroups are in particular block groups. The prototypical example of an inverse semigroup is the *symmetric inverse semigroup* I_n on the set $\{1,\dots,n\}$ which is formed by all partial one-to-one maps on $\{1,\dots,n\}$ under composition. Partial functions f, g are composed by the standard product $\circ$ for relations: $(x,y) \in f \circ g$ if there exists $z \in \{1,\dots,n\}$ such that $(x,z) \in f$ and $(z,y) \in g$. For $f \in I_n$ let $f^{-1} = \{(y,x) \ : \ (x,y) \in f\}$. Then f is regular by $f \circ f^{-1} \circ f = f$. It is easy to see that the idempotents of I_n are exactly the restrictions of the identity map on $\{1,\dots,n\}$. In particular all idempotents commute. Hence I_n is an inverse semigroup. Since finite inverse semigroups with identity, in particular I_n for $n \in \mathbb{N}$, are block groups with all their elements regular, they have polynomial-time decidable QCSP by Theorem 7.

The *full transformation semigroup* T_n is formed by all (total) transformations on $\{1,\dots,n\}$ under composition. While it is well-known and easy to check that T_n is regular, it is not a block group for $n \geq 2$. To see the latter let e, f be the constant functions on $\{1,\dots,n\}$ with image $1, 2$, respectively. Clearly e, f are idempotent and satisfy $ef = e, fe = f$; but, $e \neq f$. Hence QCSP(T_n) is NP-complete for all $n \geq 2$ by Theorem 7.

We give an easy example of a block group that is not generated by its regular elements: A semigroup $\mathbf{S}$ is a *zero semigroup* if $0 \in S$ and the multiplication is constant, that is $xy = 0$ for all $x, y \in S$. For a zero semigroup $\mathbf{S}$ the monoid $\mathbf{S}^1$ has the idempotents $0, 1$ and is a block group. However if $|S| > 1$, then $\mathbf{S}^1$ is not generated by its regular elements $0, 1$. Hence QCSP($\mathbf{S}^1$) is NP-complete for all zero semigroups $\mathbf{S}$ of size at least 2 by Theorem 7 although CSP($\mathbf{S}^1$) is in P by Theorem 4 of Bulatov, Jeavons, and Volkov. To our knowledge this is the first explicit example of an algebra $\mathbf{S}$ where CSP($\mathbf{S}$) is tractable and QCSP($\mathbf{S}$) is NP-complete.

Finally we give an example of a non-regular block group that is generated by its regular elements. Denote a transformation $f \in T_4$ by its list of images, i.e., $f = [f(1), f(2), f(3), f(4)]$. Let $\mathbf{S}$ be the transformation semigroup generated by $a = [1, 2, 2, 4]$ and $b = [4, 4, 3, 4]$. From $ab = [4, 4, 2, 4]$ and $ba = [4, 4, 4, 4]$ we see that $S = \{a, b, ab, ba\}$. We adjoin an identity to $\mathbf{S}$ to obtain the monoid $\mathbf{S}^1$. Clearly a, b and ba are idempotent, in particular, regular. Hence $\mathbf{S}^1$ is generated by regular elements. However ab is not regular because $abxab = ba$ for any $x \in S^1$. By considering the products of idempotents it is easy to check that $\mathbf{S}^1$ is a block group. Thus QCSP($\mathbf{S}^1$) is polynomial-time decidable by Theorem 7.

4 Hardness

For the hardness part of Theorem 7 we will use the following result that covers a more general setting than monoids.

▶ **Theorem 8.** *Let* $\mathbf{S}$ *be a finite semigroup that is not generated by its regular elements. The problem* QCSP($\mathbf{S}$) *is* NP*-hard. In particular, there exists a structure* $\mathbb{B}'$ *that is preserved by* $\mathbf{S}$ *and which has one* 4*-ary relation such that* QCSP($\mathbb{B}'$) *is NP-complete.*

Proof. Let J_a denote a $\mathcal{J}$-class of $\mathbf{S}$ that is maximal over all $\mathcal{J}$-classes containing an element a that is not generated by the regular elements of $\mathbf{S}$. By Lemma 1 (1), J_a is not the minimal $\mathcal{J}$-class. Then $I := \{x \in S \ : \ J_a \not\leq J_x\}$ is non-empty, and is an ideal of $\mathbf{S}$. Consider the Rees quotient $\mathbf{S}/I$. Since QCSP($\mathbf{S}/I$) $\leq_p^m$ QCSP($\mathbf{S}$) by Lemma 5, it suffices to show that QCSP($\mathbf{S}/I$) is NP-hard. So we pass from $\mathbf{S}$ to $\mathbf{S}/I$ and assume $I = \{0\}$ in the following.

Define $F := \{x \in S \ : \ J_a < J_x\}$. We have that $F \cup J_a \cup \{0\}$ is a partition of S.

By assumption, F is generated by the regular elements of $\mathbf{S}$. In particular, a is not generated by F. We claim that

$$J_a \cap \langle F \rangle = \emptyset. \tag{1}$$

We argue this claim as follows. Suppose $b \in J_a \cap \langle F \rangle$. We have $u, v \in S^1$ such that $a = ubv$. Clearly $u, v \in F$, because $J_a^2 = \{0\}$ by Lemma 1 (1). But then $a = ubv$ is generated by F, which contradicts our assumption. Hence (1) is proved.

Let $\mathbb{B}$ be the relational structure with universe $\{0, 1\}$ and with the single relation $T^{\mathbb{B}} = \{(1, 0, 0), (0, 1, 0), (0, 0, 1)$. The relational structure $\mathbb{B}$ is known to have an NP-hard CSP [22]. We claim that

$$\text{CSP}(\mathbb{B}) \leq_p^m \text{QCSP}(\mathbf{S}). \tag{2}$$

To this end, define the relational structure $\mathbb{B}'$ with universe S as follows. Set $T_b := \{(b, 0, 0), (0, b, 0), (0, 0, b)\}$ for $b \in J_a$ and

$$T'^{\mathbb{B}'} := \{(f, f, f, f) \ : \ f \in F\} \cup (0 \times S \times S \times S) \cup (\bigcup_{b \in J_a} \{b\} \times T_b).$$

By using (1) and the fact that $J_a^2 = \{0\}$, it is straightforward to verify that $T'^{\mathbb{B}'}$ is preserved by $\mathbf{S}$. We prove that CSP($\mathbb{B}$) $\leq_p^m$ QCSP($\mathbb{B}'$), which suffices to give (2).

Given an instance

$$\Phi = \exists x_1 \ldots \exists x_k \phi(x_1, \ldots, x_k) \tag{3}$$

of CSP($\mathbb{B}$) we construct an instance of QCSP($\mathbb{B}'$), as follows. Define

$$\Phi' = \forall y \exists x_1 \ldots \exists x_k \phi'(y, x_1, \ldots, x_k) \tag{4}$$

where ϕ' is obtained from ϕ by replacing every atom $T(x, x', x'')$ in ϕ by $T'(y, x, x', x'')$. The resulting instance of QCSP($\mathbb{B}'$) is Φ'.

Note that, on $\mathbb{B}'$, we have that $\phi'(s, \ldots, s)$ is true for every $s \in S \backslash J_a$. Further, when $b \in J_a$, the 0-1 assignments $f : \{x_1, \ldots, x_k\} \to \{0, 1\}$ such that $\mathbb{B}, f \models \phi(x_1, \ldots, x_k)$ correspond exactly to the 0-b assignments $f' : \{x_1, \ldots, x_k\} \to \{0, b\}$ such that $\mathbb{B}', f' \models \phi'(b, x_1, \ldots, x_k)$. Thus $\mathbb{B} \models \Phi$ if and only if $\mathbb{B}' \models \Phi'$. Hence we have (2) and that QCSP($\mathbf{S}$) is NP-hard. ◀

5 Tractability

In this section, we establish the following theorem.

▶ **Theorem 9.** *Let* $\mathbf{S}$ *be a finite monoid that is a block group and generated by its regular elements. Then* QCSP($\mathbf{S}$) *is in* P.

When $\mathbf{S}$ is a semigroup, we order assignments using the order of $\mathcal{J}$-classes of $\mathbf{S}$ as follows. Let $f, g : V \to S$ be assignments defined on the same set V of variables. We write $f \leq_{\mathcal{J}} g$ iff for all $v \in V$, it holds that $J_{f(v)} \leq J_{g(v)}$. Also, for $\mathcal{T} \in \{\mathcal{R}, \mathcal{L}\}$ we write $f \equiv_{\mathcal{T}} g$ iff for all $v \in V$, it holds that $T_{f(v)} = T_{g(v)}$.

▶ **Lemma 10** (follows from [4])**.** *Let* $\mathbf{S}$ *be a finite block group. There exists a polynomial-time algorithm that, when given as input an instance* $(\exists x_1 \dots \exists x_k \phi, \mathbb{B})$ *of* CSP($\mathbf{S}$)*, either correctly reports that there is no solution, or outputs an idempotent solution* g *such that for any solution* h *of the instance:*

- *It holds that* $g \leq_{\mathcal{J}} h$.
- *If* h *is idempotent and* $h \leq_{\mathcal{J}} g$*, then* $h = g$.

That is, the solution is $\mathcal{J}$*-minimal, and is also the unique* $\mathcal{J}$*-minimal assignment among idempotent assignments.*

Although this lemma follows from [4], we give a proof for the sake of completeness.

Proof. Let $n := |S|$. The algorithm does the following: first, it runs *arc consistency*; see [10] for a description of this algorithm. In the case that arc consistency reports *no*, the algorithm reports no; otherwise, arc consistency returns sets $A_1, \dots, A_k \subseteq S$, one for each variable $x_1, \dots, x_k$. In this latter case, it is well-known and straightforward to verify that

1. when the instance is an instance of CSP($\mathbf{A}$), each set A_i is the universe of a subalgebra of $\mathbf{A}$;
2. for any solution $f : \{x_1, \dots, x_k\} \to S$, it holds that $f(x_i) \in A_i$ (for each i); and,
3. for each atom $R(x_{i_1}, \dots, x_{i_\ell})$ of ϕ, there exists a set of tuples $U \subseteq R^{\mathbb{B}}$ such that $(A_{i_1}, \dots, A_{i_\ell}) = (\pi_1(U), \dots, \pi_\ell(U))$, where π_j is the projection onto the jth coordinate.

Let $m \geq 1$ be a natural number. As shown in [4], as $\mathbf{S}$ is a block group, the value of $(y_1^{n!} \cdots y_m^{n!})^{n!}$ depends only on the set $\{y_1, \dots, y_m\} \subseteq S$. Let $t : \wp(S) \setminus \{\emptyset\} \to S$ denote the map sending a non-empty set $\{y_1, \dots, y_m\}$ to the value $(y_1^{n!} \cdots y_m^{n!})^{n!}$. The algorithm outputs the mapping $g : \{x_1, \dots, x_k\} \to S$ given by $g(x_i) = t(A_i)$. It can be verified from item (3) that this mapping g is a solution.

Let h be any solution of the instance, and let $i \leq k$. By property (2), since $t(A_i)$ is defined as a product that involves all elements in A_i, we have that $g(x_i)$ is an element of the unique minimal $\mathcal{J}$-class of $\mathbf{S}$ that intersects A_i non-trivially and hence $J_{g(x_i)} \leq J_{h(x_i)}$. Consequently, it holds that $g \leq_{\mathcal{J}} h$. For the second claim, it suffices to show that $J_{g(x_i)} \cap A_i$ only contains one idempotent. Since $\mathbf{S}$ is a block group, the product of any pair of distinct idempotents from a $\mathcal{J}$-class lies outside of that $\mathcal{J}$-class, by Lemma 2. So, if $J_{g(x_i)} \cap A_i$ contained two distinct idempotents, their product would be an element of A_i (by property (1)) in a strictly lower $\mathcal{J}$-class of $\mathbf{S}$, a contradiction to the minimality of the $\mathcal{J}$-class $J_{g(x_i)}$ in A_i. ◀

For the rest of the section, we assume that $\mathbf{S}$ *is a finite monoid that is a block group.* We also assume that $|S| > 1$; note that Theorem 9 holds trivially in the case that $|S| = 1$. In the following, when presenting instances of CSP($\mathbf{S}$) and QCSP($\mathbf{S}$), we permit the use of atoms of the form $v = a$, where v is a variable and a is an idempotent element of $\mathbf{S}$; this is

justified by the fact that, when a is an idempotent element, the relation $\{a\}$ is preserved by the semigroup $\mathbf{S}$.

▶ **Lemma 11.** *There exists a polynomial-time algorithm that, given an instance of* $\mathrm{QCSP}_\forall(\mathbf{S})$ *having exactly one universal quantifier, computes a decision-equivalent instance of* $\mathrm{QCSP}_\forall(\mathbf{S})$ *whose sentence has the form* $\forall y \exists w_1 \dots \exists w_m \phi$*, that is, having exactly one universal quantifier that appears before the existential quantifiers.*

Proof. Let $(\Phi, \mathbb{B})$ be an instance of $\mathrm{QCSP}_\forall(\mathbf{S})$ where Φ has the form

$$\exists x_1 \dots \exists x_k \forall y \exists z_1 \dots \exists z_\ell \phi(x_1, \dots, x_k, y, z_1, \dots, z_\ell).$$

Let $T \subseteq S$. We say that a mapping $f\colon \{x_1, \dots, x_k\} \to S$ *tolerates* T if for any extension $f' : \{x_1, \dots, x_k, y\} \to S$ of f where $f'(y) \in T$, it holds that $\mathbb{B}, f' \models \exists z_1 \dots \exists z_\ell \phi$. Note that there exists an assignment that tolerates all of S if and only if $(\Phi, \mathbb{B})$ is a *yes* instance of $\mathrm{QCSP}_\forall(\mathbf{S})$. Also, note that when this condition holds, the $\mathrm{CSP}(S)$ instance $(\Phi_1, \mathbb{B})$, where

$$\Phi_1 = \exists x_1 \dots \exists x_k \exists y \exists z_1 \dots \exists z_\ell (y = 1 \wedge \phi),$$

is a *yes* instance.

The algorithm does the following. It applies the algorithm of Lemma 10 to the $\mathrm{CSP}(\mathbf{S})$ instance $(\Phi_1, \mathbb{B})$; if the result is *no*, then a *no* instance of $\mathrm{QCSP}_\forall(\mathbf{S})$ having the desired form is output. Otherwise, let h' be the resulting solution for $(\Phi_1, \mathbb{B})$, and define h to be the restriction of h' to $\{x_1, \dots, x_k\}$; the algorithm outputs the $\mathrm{QCSP}_\forall(\mathbf{S})$ instance

$$(\forall y \exists x_1 \dots \exists x_k \exists z_1 \dots \exists z_\ell (x_1 = h(x_1) \wedge \dots \wedge x_k = h(x_k) \wedge \phi), \mathbb{B}).$$

Observe that this instance is decision-equivalent to the instance

$$(\exists x_1 \dots \exists x_k \forall y \exists z_1 \dots \exists z_\ell (x_1 = h(x_1) \wedge \dots \wedge x_k = h(x_k) \wedge \phi), \mathbb{B}).$$

We argue the correctness of the algorithm by proving that, if there exists an assignment $f : \{x_1, \dots, x_k\} \to S$ that tolerates S, then the assignment h tolerates S. As f tolerates S, there exists an extension $f' : \{x_1, \dots, x_k, y, z_1, \dots, z_\ell\} \to S$ such that $f'(y) = 1$ and such that f' is a solution to the $\mathrm{CSP}(\mathbf{S})$ instance $(\Phi_1, \mathbb{B})$. We obtain that $i' = (f'h')^{|S|!}$ is also a solution to $(\Phi_1, \mathbb{B})$, where here the product is defined point-wise. As i' is an idempotent solution and has $i' \leq_\mathcal{J} h'$, we have $i' = h'$. Letting i denote the restriction of i' to $\{x_1, \dots, x_k\}$, we then have $i = h$. Since f tolerates S and h tolerates $\{1\}$, we have that fh and hence $h = i = (fh)^{|S|!}$ tolerates S. ◀

Theorem 6 and the just-established lemma allow us to restrict attention to the case of $\mathrm{QCSP}(\mathbf{S})$ where there is just one universally quantified variable, and this variable is the first (left-most) variable to appear in the quantifier prefix. We now establish two lemmas that will aid us in reasoning about such instances. Before doing so, however, we establish some terminology. Let $(\Phi = \exists y \exists z_1 \dots \exists z_\ell \phi(y, z_1, \dots, z_\ell), \mathbb{B})$ be an instance of $\mathrm{CSP}(\mathbf{S})$. Relative to such an instance, let us call an assignment $g : \{z_1, \dots, z_\ell\} \to S$ an *extension* of an element $s \in S$ if, when one takes the assignment sending the first variable y to s and extends by g, the result is a solution. When $e \in S$ is an idempotent and the algorithm of Lemma 10 returns a solution on the instance $(\exists y \exists z_1 \dots \exists z_\ell (y = e \wedge \phi), \mathbb{B})$ of $\mathrm{CSP}(\mathbf{S})$, we refer to the restriction of this solution to $\{z_1, \dots, z_\ell\}$ as the *canonical extension* of e.

▶ **Lemma 12.** *Let* $(\Phi = \exists y \exists z_1 \dots \exists z_\ell \phi(y, z_1, \dots, z_\ell), \mathbb{B})$ *be an instance of* $\mathrm{CSP}(\mathbf{S})$*. Suppose that each element of* S *has an extension, and let* $a \in S$ *be regular. By Lemma 2, there are uniquely determined idempotents* $e \in R_a, f \in L_a$*; let* g_e*,* g_f *denote their canonical extensions. Then* a *has an extension* g_a *such that* $g_e \equiv_\mathcal{R} g_a \equiv_\mathcal{L} g_f$*.*

Proof. By assumption a has some extension h. Since $eaf = a$, by Lemma 2, $g_a := g_e h g_f$ is also an extension of a with $g_a \leq_{\mathcal{J}} g_e$ and $g_a \leq_{\mathcal{J}} g_f$. Since a and e are $\mathcal{R}$-related, there exists an element $b \in J_a$ with $ab = e$. Let i be an extension of b. Then $(g_a i)^{n!}$ is an idempotent extension of e with $(g_a i)^{n!} \leq_{\mathcal{J}} g_e$. Since g_e is the unique $\mathcal{J}$-minimal extension of e by assumption, it follows that $(g_a i)^{n!} = g_e$. Together with $g_a = g_e h g_f$ this yields $g_a \equiv_{\mathcal{R}} g_e$. Similarly we obtain $g_a \equiv_{\mathcal{L}} g_f$. Thus the lemma is proved. ◀

▶ **Lemma 13.** *There exists a polynomial-time algorithm that, given a regular element $a \in S$, and an instance $(\Phi = \exists y \exists z_1 \ldots \exists z_\ell \phi(y, z_1, \ldots, z_\ell), \mathbb{B})$ of $\mathrm{CSP}(\mathbf{S})$, either reports that a has an extension, or that there exists an element without an extension.*

Proof. Let us assume that every element has an extension. Let e, f be the idempotents described in the statement of Lemma 12. For every $u \in \{1, \ldots, \ell\}$, it holds that the elements $g_e(z_u), g_f(z_u), g_a(z_u)$ are contained in the same $\mathcal{J}$-class, say, J_u. Since $\mathbf{S}$ is a block group and J_u contains a regular element (even an idempotent), by Lemma 2 we can identify J_u with $\{(i, g, j) \ : \ i, j \in I_u, g \in G_u\}$ for some index set I_u and some group $\mathbf{G}_u$. Since $g_e(z_u), g_f(z_u)$ are idempotent, they are of the form $(i_u, 1, i_u), (j_u, 1, j_u)$, respectively, for $i_u, j_u \in I$ and 1 the identity of $\mathbf{G}_u$. By Lemma 12, we have $g_u \in G_u$ such that $g_a(z_u) = (i_u, g_u, j_u)$.

Let $b \in J_a$ be such that $ab = e$. By Lemma 12, b has an extension such that $g_a g_b = g_e$, implying $g_b(z_u) = (j_u, g_u^{-1}, i_u)$. Thus we have that

$$\phi(a, (i_1, g_1, j_1), \ldots, (i_\ell, g_\ell, j_\ell)) \wedge \phi(b, (j_1, g_1^{-1}, i_1), \ldots, (j_\ell, g_\ell^{-1}, i_\ell)) \tag{5}$$

holds on $\mathbb{B}$.

We argued that if every element has an extension, then there exist elements $g_1, \ldots, g_\ell$ satisfying (5). The algorithm claimed in the lemma thus does the following. It checks to see if e and f have extensions (using Lemma 10); if not, it reports that there exists an element without an extension. Otherwise, let g_e, g_f be the canonical extensions of e and f, respectively, and let b be as described. The groups $\mathbf{G}_1, \ldots, \mathbf{G}_\ell$ and the indices $i_1, \ldots, i_\ell, j_1, \ldots, j_\ell$ are uniquely determined by g_e and g_f.

We will argue that determining the existence of elements $g_1, \ldots, g_\ell$ as in (5) is a CSP over a coset generating operation, i.e., its relations are closed under the operation $xy^{-1}z$. This CSP can be solved in polynomial time by Theorem 33 of [13], in particular, by using the known fact (which is straightforwardly verified) that a relation closed under the operation $xy^{-1}z$ of a group is a coset of a subgroup of the group. (Note that Theorem 33 of [13] deals with a single group $\mathbf{G}$; it can be employed for our purposes here by simply taking $\mathbf{G}$ to be the product of all groups $\mathbf{G}_i$ that may arise, and then identifying an element g of a group $\mathbf{G}_i$ with the element of $\mathbf{G}$ equal to g at the coordinate corresponding to $\mathbf{G}_i$, and as equal to the identity element of the respective group everywhere else.) This argument will thus conclude the proof of the Lemma; if such elements $g_1, \ldots, g_\ell$ exist, then a has an extension, otherwise some element does not have an extension.

Let ψ be an arbitrary constraint of ϕ, and assume that, for $u \in \{1, \ldots, \ell\}$, it holds that $r_u, s_u, t_u \in G_u$ are such that, on $\mathbb{B}$, the following hold:

$$\psi(a, (i_1, r_1, j_1), \ldots, (i_\ell, r_\ell, j_\ell)), \psi(a, (i_1, s_1, j_1), \ldots, (i_\ell, s_\ell, j_\ell)), \psi(a, (i_1, t_1, j_1), \ldots, (i_\ell, t_\ell, j_\ell)).$$

It follows that, on $\mathbb{B}$, the following hold:

$$\psi(a, (i_1, r_1, j_1), \ldots, (i_\ell, r_\ell, j_\ell)), \psi(b, (j_1, s_1^{-1}, i_1), \ldots, (j_\ell, s_\ell^{-1}, i_\ell)), \psi(a, (i_1, t_1, j_1), \ldots, (i_\ell, t_\ell, j_\ell)).$$

Since ψ is preserved by the semigroup multiplication and $aba = a$ by Lemma 2 (3), we obtain that

$$\psi(a, (i_1, r_1 s_1^{-1} t_1, j_1), \ldots, (i_\ell, r_\ell s_\ell^{-1} t_\ell, j_\ell))$$

holds on $\mathbb{B}$. Similarly, we have $bab = b$ which yields that

$$\psi(b, (i_1, t_1^{-1}s_1r_1^{-1}, j_1), \ldots, (i_\ell, t_\ell^{-1}s_\ell r_\ell^{-1}, j_\ell))$$

holds on $\mathbb{B}$. So the induced constraints are closed under $xy^{-1}z$. ◀

Proof of Theorem 9. Let $\mathbf{S}$ be a monoid that is a block group and generated by its regular elements. We have that CSP($\mathbf{S}$) is polynomial-time decidable by Theorem 4. Thus, by Theorem 6 and Lemma 11, it suffices to prove polynomial-time decidability of the restriction of QCSP$_\forall$($\mathbf{S}$) to instances with exactly one universal quantifier that appears before the existential quantifiers. Let $I = (\Psi = \forall y \exists z_1 \ldots \exists z_\ell \phi, \mathbb{B})$ be such an instance, and define I' to be the instance $(\Phi = \exists y \exists z_1 \ldots \exists z_\ell \phi, \mathbb{B})$ of CSP($\mathbf{S}$). The algorithm first checks that each idempotent has an extension with respect to I'; if this is not the case, the algorithm returns *no*. Since $\mathbf{S}$ is generated by its regular elements, I is a *yes* instance if and only if each regular element $a \in S$ has an extension with respect to I' (recall Proposition 3). Checking the latter condition can be done by looping over each regular element $a \in S$ and invoking the algorithm of Lemma 13. ◀

6 Conclusion

We investigated the complexity of quantified constraint satisfaction problems from monoids. While QCSP($\mathbb{B}$) for an arbitrary relational structure may be PSPACE-complete, QCSP($\mathbf{S}$) for a monoid $\mathbf{S}$ is always in NP. In our main result Theorem 7 we established a dichotomy between tractable and NP-complete QCSP for all finite monoids via some simple algebraic conditions. Note for any semilattice $\mathbf{S}$ without unit Börner et al showed that QCSP($\mathbf{S}$) is PSPACE-complete [2, Theorem 6.1]. A complete characterization of the complexity of QCSP for all semigroups remains open.

Combining the results of [4, 2] and the present paper we can compare the complexity of constraint satisfaction and quantified constraint satisfaction for the same fixed semigroup $\mathbf{S}$. We observe the following behavior:

1. **a.** CSP($\mathbf{S}$) in P and QCSP($\mathbf{S}$) in P if $\mathbf{S}$ is a block group, a monoid and generated by its regular elements;
   b. CSP($\mathbf{S}$) in P and QCSP($\mathbf{S}$) is NP-complete if $\mathbf{S}$ is a block group, a monoid and not generated by its regular elements;
   c. CSP($\mathbf{S}$) in P and QCSP($\mathbf{S}$) is PSPACE-complete if $\mathbf{S}$ is a semilattice without 1;
2. **b.** CSP($\mathbf{S}$) is NP-complete and QCSP($\mathbf{S}$) is NP-complete if $\mathbf{S}$ is not a block group but a monoid;
   c. CSP($\mathbf{S}$) is NP-complete and QCSP($\mathbf{S}$) is PSPACE-complete if $\mathbf{S}$ is not a block group but has a semilattice without 1 as homomorphic image.

The case (2a) that CSP($\mathbf{S}$) is NP-complete and QCSP($\mathbf{S}$) is in P cannot occur and is omitted. The cases above can be attained by concrete monoids given in Section 3 except for examples witnessing PSPACE-completeness. Note that clearly (1c) occurs because semilattices without 1 exist. For (2c) we consider an idempotent semigroup $\mathbf{S}$ with elements a, b, e, f such that

$$a^2 = a, ax = e \text{ for all } x \neq a,$$
$$b^2 = b, bx = f \text{ for all } x \neq b,$$
$$ex = e, fx = f \text{ for all } x \in S.$$

Then $ef = e \neq f = fe$ implies that $\mathbf{S}$ is not a block group. Further $I = \{e, f\}$ is an ideal with quotient $\mathbf{S}/I$ isomorphic to a 3-element semilattice without 1. Thus $\mathbf{S}$ witnesses case (2c).

References

1 Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):Art. 3, 19, 2014. doi:10.1145/2556646.

2 Ferdinand Börner, Andrei A. Bulatov, Hubie Chen, Peter Jeavons, and Andrei A. Krokhin. The complexity of constraint satisfaction games and QCSP. *Inform. and Comput.*, 207(9):923–944, 2009. doi:10.1016/j.ic.2009.05.003.

3 Andrei Bulatov, Peter Jeavons, and Andrei Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.

4 Andrei Bulatov, Peter Jeavons, and Mikhail Volkov. Finite semigroups imposing tractable constraints. In *Semigroups, algorithms, automata and languages (Coimbra, 2001)*, pages 313–329. World Sci. Publ., River Edge, NJ, 2002. doi:10.1142/9789812776884_0011.

5 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006. doi:10.1145/1120582.1120584.

6 Catarina Carvalho, Florent R. Madelaine, and Barnaby Martin. From complexity to algebra and back: Digraph classes, collapsibility, and the PGP. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 462–474, 2015.

7 Hubie Chen. The complexity of quantified constraint satisfaction: collapsibility, sink algebras, and the three-element case. *SIAM J. Comput.*, 37(5):1674–1701, 2008. doi:10.1137/060668572.

8 Hubie Chen. Quantified constraint satisfaction and the polynomially generated powers property. *Algebra Universalis*, 65(3):213–241, 2011. doi:10.1007/s00012-011-0125-4.

9 Hubie Chen. Meditations on quantified constraint satisfaction. In *Logic and program semantics*, volume 7230 of *Lecture Notes in Comput. Sci.*, pages 35–49. Springer, Heidelberg, 2012. doi:10.1007/978-3-642-29485-3_4.

10 Hubie Chen, Victor Dalmau, and Berit Grußien. Arc consistency and friends. *J. Logic Comput.*, 23(1):87–108, 2013. doi:10.1093/logcom/exr039.

11 Hubie Chen, Florent Madelaine, and Barnaby Martin. Quantified constraints and containment problems. *Log. Methods Comput. Sci.*, 11(3):3:9, 28, 2015.

12 Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. doi:10.1137/1.9780898718546.

13 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104 (electronic), 1999. doi:10.1137/S0097539794266766.

14 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *J. Combin. Theory Ser. B*, 48(1):92–110, 1990. doi:10.1016/0095-8956(90)90132-J.

15 John M. Howie. *Fundamentals of semigroup theory*, volume 12 of *London Mathematical Society Monographs. New Series*. The Clarendon Press, Oxford University Press, New York, 1995. Oxford Science Publications.

16 Paweł Idziak, Petar Marković, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010. doi:10.1137/090775646.

17 Florent Madelaine and Barnaby Martin. QCSP on partially reflexive cycles – the wavy line of tractability. In *Computer science – theory and applications*, volume 7913 of *Lecture Notes in Comput. Sci.*, pages 322–333. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-38536-0_28.

18 Florent R. Madelaine and Barnaby Martin. On the complexity of the model checking problem. *CoRR*, abs/1210.6893, 2012.

19 Barnaby Martin. QCSP on partially reflexive forests. In *Principles and Practice of Constraint Programming – CP 2011 – 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, pages 546–560, 2011.

20 Barnaby Martin and Florent Madelaine. Towards a trichotomy for quantified H-coloring. In *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe (CiE)*, pages 342–352, 2006.

21 Peter Mayr. On finitely related semigroups. *Semigroup Forum*, 86(3):613–633, 2013. doi:10.1007/s00233-012-9455-6.

22 Thomas J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978.

Non-Homogenizable Classes of Finite Structures*

Albert Atserias[1] and Szymon Toruńczyk[2]

1 Universitat Politècnica de Catalunya, Spain
2 University of Warsaw, Poland

Abstract

Homogenization is a powerful way of taming a class of finite structures with several interesting applications in different areas, from Ramsey theory in combinatorics to constraint satisfaction problems (CSPs) in computer science, through (finite) model theory. A few sufficient conditions for a class of finite structures to allow homogenization are known, and here we provide a necessary condition. This lets us show that certain natural classes are not homogenizable: 1) the class of locally consistent systems of linear equations over the two-element field or any finite Abelian group, and 2) the class of finite structures that forbid homomorphisms from a specific MSO-definable class of structures of treewidth two. In combination with known results, the first example shows that, up to pp-interpretability, the CSPs that are solvable by local consistency methods are distinguished from the rest by the fact that their classes of locally consistent instances are homogenizable. The second example shows that, for MSO-definable classes of forbidden patterns, treewidth one versus two is the dividing line to homogenizability.

1998 ACM Subject Classification G.2 Discrete mathematics

Keywords and phrases Fraïssé class, amalgmation class, reduct, Constraint Satisfaction Problem, bounded width

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.16

1 Introduction

A relational structure with a countable domain is called homogeneous if it is highly symmetric in the precise technical sense that any isomorphism between any two of its finite induced substructures extends to an automorphism of the whole structure. In many areas of combinatorics, logic, discrete geometry, and computer science, homogeneous structures abound, often in the form of nicely behaved limit objects for classes of finite structures. Typical examples include the Rado graph $\mathcal{R}$, which can be seen as the limit of the class of all finite graphs; the linear order of the rational numbers $\mathcal{Q}$, seen as the limit of all finite linear orders; or the countable Urysohn space $\mathcal{U}$, the limit of all rational metric spaces. The literature on the subject is very extensive; we refer the reader to [16] for a recent survey.

Homogeneous structures arise as limits of well-behaved classes of finite structures, in a way made precise by Fraïssé's theorem, which describes them combinatorially in a finitary manner. It states that a homogeneous structure is characterized, up to isomorphism, by its age, i.e., the class of its finite induced substructures. Moreover, classes of finite structures arising as ages of homogeneous structures are precisely Fraïssé classes, i.e., classes closed under taking induced substructures and under amalgamation – a form of glueing pairs of structures along a common induced substructure (see [13] and Section 2 for precise definitions).

* Partly funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ERC-2014-CoG 648276 AUTAR).

© Albert Atserias and Szymon Toruńczyk;
licensed under Creative Commons License CC-BY

Thanks to Fraïssé's theorem, combinatorial arguments involving finite structures can often be replaced by, or aided by, arguments involving highly symmetric, infinite structures. In combinatorics, for example, homogeneous structures appear unavoidably in structural Ramsey theory [17]. At the intersection between combinatorics and computer science, homogeneous structures appear in the theory of logical limit laws for various models of random graphs [15]. In computer science proper, homogeneous structures appear in the theory of constraint satisfaction problems [5], automata theory [6], and verification [7].

One of the advantages of working with homogeneous structures, rather than classes of finite structures, is that their automorphism groups are very rich. For example, over a finite relational signature, the homogeneity of the structure immediately implies that, up to automorphism, it has finitely many elements, pairs of elements, triples, etc. In model theoretic terms, this means that the structure is ω-categorical by the classical Ryll-Nardzewski theorem, and its first-order theory admits elimination of quantifiers. In turn, since in any such structure there are only finitely many first-order definable relations of each arity, homogeneous structures over finite relational signatures are, in a strong technical way, close to being finite.

Thus, with Fraïssé's theorem in hand and the many applications of homogeneous structures in mind, it becomes quite important a task to identify more Fraïssé classes. More generally, one would like to identify classes of finite structures that are perhaps not Fraïssé classes themselves, but appear as reducts of some Fraïssé class over a richer yet finite signature. Such classes of finite structures are called *homogenizable* [9]. The point in case is that the lifted Fraïssé class can be thought of as taming its reduct by providing a homogeneous structure that plays the role of limit object for it. Many of the application examples mentioned above do actually go through lifted Fraïssé classes and their corresponding homogeneous Fraïssé limits. See [14] and the references therein for a discussion on this.

A noticeable amount of work has gone into providing sufficient conditions for a class of finite structures to be homogenizable. Instances include the model-theoretic methods of Covington [9], and the combinatorial explicit constructions of Hubička and Nešetřil [14]. Here we provide a combinatorial necessary condition for homogenizability (Theorem 3.2 in Section 3). This allows us to prove that certain natural classes of finite structures previously considered in the literature are not homogenizable.

Our first example of a non-homogenizable class comes from the theory of constraint satisfaction problems (CSPs). We show that the class of locally consistent systems of linear equations over the two-element field is not homogenizable. More generally, the result holds for systems of equations over any finite Abelian group. This answers a question first raised by the first author of this paper in [2]. Precisely, by a locally consistent system of equations we mean one whose satisfiability cannot be refuted by the (j, k)-consistency algorithm for small j and k, which is a well-studied heuristic algorithm for solving CSPs. Moreover, in combination with the resolution of the Bounded Width Conjecture by Barto and Kozik [4], this shows that the constraint languages whose classes of locally consistent instances are homogenizable are, up to pp-interpretability, precisely those that are solvable by local consistency methods. All this is worked out in Section 4.

In Section 5 we give a second example of a non-homogenizable class that, in this case, is motivated by the works of Hubička and Nešetřil [14], and Erdös, Tardif, and Tardos [10]. It was shown in [14] that every class of finite structures that is of the form $\mathrm{Forb}_h(\mathcal{F})$, where $\mathcal{F}$ is a *regular* class of connected finite structures, is homogenizable. In words, $\mathrm{Forb}_h(\mathcal{F})$ is the class of finite structures that do not admit homomorphisms from any structure in $\mathcal{F}$. The notion of regularity considered in [14] is closely related to the notion of regularity in automata theory, and agrees with it on coloured paths and trees. However, our second example shows

that even if $\mathcal{F}$ is MSO-definable and has maximum treewidth two, the class $\mathrm{Forb}_h(\mathcal{F})$ need not be homogenizable. This shows that for MSO-definable classes, treewidth one versus two of the forbidden structures in $\mathcal{F}$ is the dividing line to homogenizability.

2 Preliminaries

Signatures, structures, reducts, and expansions

A relational signature Σ is a set of relation symbols $R_1, R_2, \ldots$, each with an associated natural number called its arity. In this paper, we consider only finite relational signatures. A Σ-structure $\mathbb{A} = (A; R_1^{\mathbb{A}}, R_2^{\mathbb{A}}, \ldots)$ is composed of a set A, called its domain, and a relation $R^{\mathbb{A}} \subseteq A^k$ on A for each R in Σ, where k is the arity of R. We say that $R^{\mathbb{A}}$ is the interpretation of R in $\mathbb{A}$. A Σ-structure is sometimes referred to as a structure over the signature Σ. If Σ^+ is a signature that contains Σ and $\mathbb{A}^+$ is a Σ^+-structure, then the Σ-reduct of $\mathbb{A}^+$ is the structure $\mathbb{A}$ obtained from $\mathbb{A}^+$ by forgetting all relations from $\Sigma^+ - \Sigma$. In this case, we also say that $\mathbb{A}^+$ is an expansion of $\mathbb{A}$. Expansions and reducts are also called lifts and shadows, respectively.

Substructures, homomorphisms, and embeddings

If $\mathbb{A}$ is a Σ-structure and X is a subset of its domain A, we write $\mathbb{A}[X]$ for the substructure of $\mathbb{A}$ induced by X, that is, the Σ-structure with domain X in which each relation symbol R in Σ is interpreted by $R^{\mathbb{A}} \cap X^k$, where k is the arity of R.

Let $\mathbb{A}$ and $\mathbb{B}$ be structures over the same relational signature Σ. Let A and B denote their domains. A homomorphism from $\mathbb{A}$ to $\mathbb{B}$ is a mapping $f : A \to B$ for which the inclusion $f(R^{\mathbb{A}}) \subseteq R^{\mathbb{B}}$ holds for every R in Σ. The homomorphism is strong if in addition the inclusion $f(A^k - R^{\mathbb{A}}) \subseteq B^k - R^{\mathbb{B}}$ holds for every R in Σ, where k is the arity of R. A monomorphism from $\mathbb{A}$ to $\mathbb{B}$ is an injective homomorphism. Whenever A is a subset of B and the inclusion $A \to B$ is a monomorphism, we say that $\mathbb{A}$ is a substructure of $\mathbb{B}$. An embedding from $\mathbb{A}$ to $\mathbb{B}$ is an injective strong homomorphism. Whenever A is a subset of B and the inclusion $A \to B$ is an embedding, we say that $\mathbb{A}$ is an induced substructure of $\mathbb{B}$. An isomorphism from $\mathbb{A}$ to $\mathbb{B}$ is a surjective embedding. If there is an isomorphism from $\mathbb{A}$ to $\mathbb{B}$ we say that the two structures are isomorphic. If $f : A \to B$ is a partial mapping with domain $X \subseteq A$ and image $Y \subseteq B$, we say that f is a partial homomorphism from $\mathbb{A}$ to $\mathbb{B}$ if it is a homomorphism from $\mathbb{A}[X]$ to $\mathbb{B}[Y]$. We write $\binom{\mathbb{B}}{\mathbb{A}}$ to denote the set of all embeddings from $\mathbb{A}$ to $\mathbb{B}$.

Amalgamation

If $\mathbb{B}$ and $\mathbb{C}$ are Σ-structures with domains B and C, we write $\mathbb{B} \cup \mathbb{C}$ for their union, i.e. the Σ-structure with domain $B \cup C$ and relations $R^{\mathbb{B}\cup\mathbb{C}} = R^{\mathbb{B}} \cup R^{\mathbb{C}}$ for every R in Σ. Let f and g be embeddings from the same structure $\mathbb{A}$ into structures $\mathbb{B}$ and $\mathbb{C}$, respectively. The structure $\mathbb{D}$ is an amalgam of $\mathbb{B}$ and $\mathbb{C}$ through f and g if there exist embeddings f' and g' from $\mathbb{B}$ to $\mathbb{D}$ and $\mathbb{C}$ to $\mathbb{D}$, respectively, such that the diagram in Figure 1 commutes, i.e., $f' \circ f = g' \circ g$.

We say that $\mathbb{D}$ is a strong amalgam if $f'(B) \cap g'(C) = (f' \circ f)(A) = (g' \circ g)(A)$, where A, B and C denote the domains of $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$, respectively. We say that $\mathbb{D}$ is a free amalgam if it is strong and, additionally, $\mathbb{D} = \mathbb{D}[f'(B)] \cup \mathbb{D}[g'(C)]$. We also say that $\mathbb{D}$ is the union of $\mathbb{B}$ and $\mathbb{C}$ amalgamated along $\mathbb{A}$ through f and g via f' and g'. Note that the free amalgam of $\mathbb{B}$ and $\mathbb{C}$ through f and g is uniquely defined up to isomorphism, and is isomorphic to the disjoint union of $\mathbb{B}$ and $\mathbb{C}$, quotiented by the equivalence relation identifying $f(x)$ with $g(x)$,

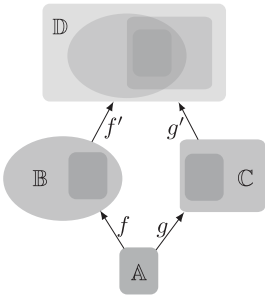


Figure 1 Amalgamation of $\mathbb{B}$ and $\mathbb{C}$ through f and g. All mappings are embeddings.

for $x \in \mathbb{A}$. We denote this free amalgam $f \cup_{\mathbb{A}} g$. When f and g are implicit, we denote it $\mathbb{B} \cup_{\mathbb{A}} \mathbb{C}$. We also say that $\mathbb{B}$ and $\mathbb{C}$ are glued along $\mathbb{A}$.

Classes of structures

All our structures will have finite or countably infinite domain. Moreover we assume that all structures have a domain that is a subset of a common background countable set, say $\mathbb{N}$. For a fixed signature Σ, a class of structures is a set of structures that is closed under isomorphisms, i.e. if $\mathbb{A}$ and $\mathbb{B}$ are isomorphic structures and $\mathbb{A}$ belongs to the class, then $\mathbb{B}$ also belongs to the class. A class of structures $\mathcal{C}$ is closed under amalgamation if for every two embeddings f and g from the same structure $\mathbb{A}$ in $\mathcal{C}$ into structures $\mathbb{B}$ and $\mathbb{C}$ in $\mathcal{C}$, there exists in $\mathcal{C}$ an amalgam of $\mathbb{B}$ and $\mathbb{C}$ through f and g. A class of finite structures is an amalgamation class, also called a Fraïssé class, if it is closed under taking induced substructures and amalgamation. For example, the class of all finite graphs is an amalgamation class – in fact, it is closed under free amalgamation – so is the class of all finite digraphs. The class of all finite linear orders is also an amalgamation class, although it is not closed under free amalgamation. Fraïssé's theorem states that a class is Fraïssé if and only if it is the class of finite induced substructures of a homogeneous structure.

For two signatures Σ and Σ^+ with the second containing the first, if $\mathcal{C}$ and $\mathcal{C}^+$ are classes of Σ-structures and Σ^+-structures, respectively, then we say that $\mathcal{C}$ is the Σ-reduct of $\mathcal{C}^+$ if $\mathcal{C}$ is the class of Σ-reducts of the structures in $\mathcal{C}^+$.

Homogenizable classes

We say that a class of Σ-structures is *homogenizable* if there is a signature Σ^+ extending Σ, and an amalgamation class $\mathcal{C}^+$ over Σ^+, such that $\mathcal{C}$ is the Σ-reduct of $\mathcal{C}^+$. For a class of Σ-structures $\mathcal{F}$, let $\mathrm{Forb}_h(\mathcal{F})$ denote the class of all finite Σ-structures $\mathbb{A}$ such that for no $\mathbb{F}$ in $\mathcal{F}$ there is a homomorphism from $\mathbb{F}$ to $\mathbb{A}$. Hubička and Nešetřil define a notion of regularity, which we call HN-regularity (we omit its technical definition), and prove in Theorem 3.1 from [14] that if $\mathcal{F}$ is a HN-regular class of finite connected structures, then $\mathrm{Forb}_h(\mathcal{F})$ is homogenizable. In particular, if $\mathcal{F}$ is finite, then $\mathrm{Forb}_h(\mathcal{F})$ is homogenizable.

▶ **Example 2.1.** Let Σ be the signature that consists of one binary predicate $\vec{E}$ and two unary predicates S and T. Let $\mathbb{P}_n$ denote a simple directed $\vec{E}$-path of length n from a unique S-colored node to a unique T-colored node. The class $\mathcal{F} = \{\mathbb{P}_n : n \geq 0\}$ is HN-regular, and therefore, by [14], the class $\mathrm{Forb}_h(\mathcal{F})$ is homogenizable. It consists of digraphs whose nodes are possibly labeled with S or T, and there is no directed path from an S-labeled node to a T-labeled node. We show that $\mathrm{Forb}_h(\mathcal{F})$ is homogenizable by a direct construction. Let Σ^+

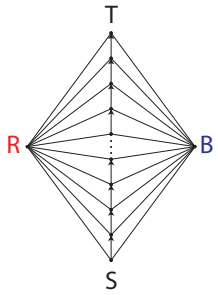


Figure 2 Forbidden structure $\mathbb{F}_n$.

be the extension of Σ by two unary predicates I and O. Let $\mathcal{C}^+$ consist of all Σ^+-structures $\mathbb{A}^+$ such that the domain of $\mathbb{A}^+$ is partitioned into $I^{\mathbb{A}^+}$ and $O^{\mathbb{A}^+}$, and that $S^{\mathbb{A}^+} \subseteq I^{\mathbb{A}^+}$, $T^{\mathbb{A}^+} \subseteq O^{\mathbb{A}^+}$, and there are no $\vec{E}$-edges starting in $I^{\mathbb{A}^+}$ and ending in $O^{\mathbb{A}^+}$. Then $\mathcal{C}^+$ is an amalgamation class, as it is closed under free amalgamation. The class $\mathrm{Forb}_h(\mathcal{F})$ is the Σ-reduct of $\mathcal{C}^+$: a structure $\mathbb{A}$ in $\mathrm{Forb}_h(\mathcal{F})$ extends to a structure $\mathbb{A}^+$ in $\mathcal{C}^+$, in which $I^{\mathbb{A}^+}$ is the set of vertices reachable from $S^{\mathbb{A}}$ by a directed $\vec{E}$-path, and $O^{\mathbb{A}^+}$ is its complement. ◀

3 Necessary condition for homogenizability

Fix a finite relational signature Σ. In this section all structures are over this signature, or over a signature Σ^+ that extends Σ. Before we state the necessary condition for homogenizability we need some notation and terminology.

Let $\mathcal{C}$ be a class of finite structures. If $\mathbb{A}$, $\mathbb{L}$ and $\mathbb{R}$ are structures in $\mathcal{C}$, and $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$ are embeddings such that no amalgam of $\mathbb{L}$ and $\mathbb{R}$ through L and R is in $\mathcal{C}$, then we say that $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ is a *diagram that witnesses failure* of amalgamation of $\mathcal{C}$. We illustrate the definitions with a running example.

▶ **Example 3.1** (Running example)**.** Let $\mathbb{F}_n$ denote the structure depicted in Figure 2, with n vertices in the middle column.

The signature Σ of this structure consists of one binary predicate E, the undirected edges, one binary predicate $\vec{E}$, the vertical directed edges, and four unary predicates R (for *red*), B (for *blue*), S (for *source*), and T (for *target*), each appearing in the structure exactly once. Observe that the colours S and T ensure that $\mathcal{F}$ is an antichain in the homomorphism pre-order, i.e. there are no homomorphisms from $\mathbb{F}_n$ to $\mathbb{F}_m$ if $n \neq m$. Let $\mathcal{C} = \mathrm{Forb}_h(\mathcal{F})$. In the running example, we will demonstrate that the class $\mathcal{C}$ is not homogenizable.

Choose a large natural number n. Let $\mathbb{L}$ denote the left part of the structure $\mathbb{F}_n$ obtained by removing the blue vertex (labeled B). Symmetrically, let $\mathbb{R}$ denote the right part of $\mathbb{F}_n$ obtained by removing the red vertex (labeled R). Let $\mathbb{A}$ denote the intersection of $\mathbb{L}$ and $\mathbb{R}$, i.e., the $\vec{E}$-path with n vertices starting at the S-labeled vertex and ending at the T-labeled vertex. Let $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$ be the inclusion mappings. Then any amalgamation of L and R necessarily is a homomorphic image of $\mathbb{F}_n$. Hence $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ is a diagram that witnesses failure of amalgamation of $\mathcal{C}$. ◀

Let $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ be a diagram that witnesses failure of amalgamation of $\mathcal{C}$. For a structure $\mathbb{J}$ and a partial mapping $C : \binom{\mathbb{J}}{\mathbb{A}} \to \{L, R\}$, let $\mathbb{J}^C$ be the structure that is obtained by glueing to $\mathbb{J}$, for each π in $\mathrm{Dom}(C)$, a fresh copy of either $\mathbb{L}$ or $\mathbb{R}$ depending on whether $C(\pi) = L$ or $C(\pi) = R$. More formally, $\mathbb{J}^C$ is defined by induction on the cardinality

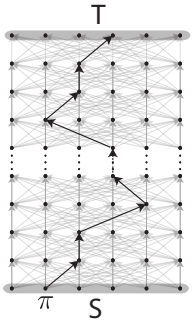


Figure 3 The structure $\mathbb{A} \otimes m$, with an embedding $\pi \in \mathcal{E}_{\mathbb{A},m}$.

of the domain of C: if $\mathrm{Dom}(C) = \emptyset$, then $\mathbb{J}^C = \mathbb{J}$; otherwise, if $C = C' \cup \{\pi \mapsto \sigma\}$, where $\pi \in \binom{\mathbb{J}}{\mathbb{A}}$ and $\sigma \in \{L, R\}$, then define $\mathbb{J}^C = \pi' \cup_{\mathbb{A}} \sigma$, where $\pi' : \mathbb{A} \to \mathbb{J}^{C'}$ is $\pi : \mathbb{A} \to \mathbb{J}$ composed with the identity embedding from $\mathbb{J}$ to $\mathbb{J}^{C'}$.

For a natural number m and a Σ-structure $\mathbb{A}$ with domain A, let $\mathbb{A} \otimes m$ denote the structure with domain $A \times [m]$ in which the interpretation of a relation R in Σ of arity k is the set of all tuples $((a_1, i_1), (a_2, i_2), \ldots, (a_k, i_k))$ where $(a_1, \ldots, a_k) \in R^{\mathbb{A}}$ and $i_1, \ldots, i_k \in [m]$. Observe that every function $f : A \to [m]$ induces an embedding $\pi_f : \mathbb{A} \to \mathbb{A} \otimes m$, defined by $\pi_f(a) = (a, f(a))$. Let $\mathcal{E}_{\mathbb{A},m}$ denote the set of all embeddings of the form π_f for $f : A \to [m]$. In particular, $\mathcal{E}_{\mathbb{A},m}$ is a subset of $\binom{\mathbb{A}\otimes m}{\mathbb{A}}$ containing exactly $m^{|A|}$ embeddings.

A diagram $L : \mathbb{A} \to \mathbb{L},\ R : \mathbb{A} \to \mathbb{R}$ is *confusing* for $\mathcal{C}$ if the following conditions hold:

1. it witnesses failure of amalgamation of $\mathcal{C}$, and
2. for every natural number m, if $\mathbb{J} = \mathbb{A} \otimes m$, then for every coloring $C : \mathcal{E}_{\mathbb{A},m} \to \{L, R\}$ the structure $\mathbb{J}^C$ belongs to the class $\mathcal{C}$.

Its *order* is the cardinality of the domain of $\mathbb{A}$.

▶ **Theorem 3.2.** *If $\mathcal{C}$ is a homogenizable class of finite structures, then there exists a natural number r such that every confusing diagram for $\mathcal{C}$ has order at most r.*

This theorem is the main technical result of this paper. Before we prove it, we illustrate it by applying it to our running example.

▶ **Example 3.3.** Fix natural numbers m and n. Let $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$ be defined as in Example 3.1. The structure $\mathbb{J} = \mathbb{A} \otimes m$ is depicted in Figure 3.

Its domain is $[n] \times [m]$, and every element $(i, j) \in [n] \times [m]$ with $i \leq n - 1$ is connected by an $\vec{E}$-edge to every element $(i + 1, k) \in [n] \times [m]$. The embeddings $\mathcal{E}_{\mathbb{A},m}$ correspond to functions $f : [n] \to [m]$. If $C : \mathcal{E}_{\mathbb{A},m} \to \{L, R\}$ is a coloring, then $\mathbb{J}^C$ is obtained by considering all functions $f : [n] \to [m]$, and connecting every vertex along the the path $\{(i, f(i)) : 1 \leq i \leq n\}$ to a fresh vertex which is red if $C(f) = L$, and blue if $C(f) = R$. Observe that no structure $\mathbb{F}$ in $\mathcal{F}$ maps homomorphically to $\mathbb{J}^C$. Therefore, $\mathbb{J}^C$ belongs to $\mathcal{C} = \mathrm{Forb}_h(\mathcal{F})$. Since m is arbitrary, this shows that the diagram $L : \mathbb{A} \to \mathbb{L},\ R : \mathbb{A} \to \mathbb{R}$ is confusing for $\mathcal{C}$. Since its order is $|A| = n$, and n is arbitrary, Theorem 3.2 implies that $\mathcal{C}$ is not a reduct of any amalgamation class. ◀

Theorem 3.2 follows easily from Lemma 3.4 stated below.

Let $L : \mathbb{A} \to \mathbb{L}, R : \mathbb{A} \to \mathbb{R}$ witness failure of amalgamation of $\mathcal{C}$. An (L, R)-*confusion* for $\mathcal{C}$ is a structure $\mathbb{J}$ in $\mathcal{C}$, together with a set $\mathcal{E} \subseteq \binom{\mathbb{J}}{\mathbb{A}}$, such that $\mathbb{J}^C$ is in $\mathcal{C}$ for every coloring $C : \mathcal{E} \to \{L, R\}$. For $\mathcal{E} \subseteq \binom{\mathbb{J}}{\mathbb{A}}$ and a natural number r bounded by the cardinality of the

domain of $\mathbb{A}$, let $\mathcal{E}_r$ denote the set of all restrictions $\pi|_X$ of π in $\mathcal{E}$, where X ranges over all r-element subsets of the domain of $\mathbb{A}$.

▶ **Lemma 3.4.** *Let r and t be natural numbers, and let $\mathcal{C}$ be a class of Σ-structures. There exist numbers p and q (depending on r and t only) such that the following condition implies that $\mathcal{C}$ is not a reduct of any amalgamation class over a signature with at most t predicates of arity at most r:*

there is a diagram $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ that witnesses failure of amalgamation of $\mathcal{C}$ and of order at least r, and there is an (L, R)-confusion $(\mathbb{J}, \mathcal{E})$ for $\mathcal{C}$ satisfying

$$|\mathcal{E}| > p \cdot |\mathcal{E}_r| + q^{\binom{|\mathbb{A}|}{r}}. \tag{1}$$

First we show how Theorem 3.2 follows from Lemma 3.4.

Proof of Theorem 3.2. Suppose that $\mathcal{C}$ has confusing diagrams of arbitrarily large order. For every two fixed natural numbers r and t, we apply Lemma 3.4 to conclude that $\mathcal{C}$ is not a reduct of an amalgamation class over a signature with t symbols of arity at most r. Let p and q be as in the statement of the lemma. Consider a confusing diagram $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ and let n be its order. Fix a natural number m, and let $\mathbb{J} = \mathbb{A} \otimes m$ and $\mathcal{E} = \mathcal{E}_{\mathbb{A},m}$. Then $(\mathbb{J}, \mathcal{E})$ is an (L, R)-confusion for $\mathcal{C}$, by the definition of confusing diagram, and $|\mathcal{E}| = m^n$ and $|\mathcal{E}_r| = m^r$. Since the order n of the diagram can be chosen arbitrarily large, we can assume $n > r$. Taking m large enough, so that $p \cdot m^{n-1} > q^{\binom{n}{r}}$ and $m > 2p$, we get:

$$p \cdot |\mathcal{E}_r| + q^{\binom{|\mathbb{A}|}{r}} = p \cdot m^r + q^{\binom{n}{r}} \leq p \cdot m^{n-1} + p \cdot m^{n-1} < m^n = |\mathcal{E}|,$$

which gives condition (1) in Lemma 3.4. Since t and r were arbitrary, this proves that $\mathcal{C}$ is not the reduct of an amalgamation class. ◀

It remains to prove the lemma.

Proof of Lemma 3.4. Fix natural numbers r and t. In anticipation of the proof, let q be the maximum number of atomic types of r-tuples over any signature with at most t predicates of arity at most r, and let $p = \lceil \log_2(q) \rceil$. Suppose that $\mathcal{C}$ is a class of Σ-structures as in the lemma, with a diagram $L : \mathbb{A} \to \mathbb{L}$, $R : \mathbb{A} \to \mathbb{R}$ that witnesses its failure of amalgamation, and an (L, R)-confusion $(\mathbb{J}, \mathcal{E})$ satisfying condition (1) from Lemma 3.4.

Let $\mathbb{B}^+$ be a Σ^+-structure with domain B and let $f : A \to B$ be a function from some set A to B. Define the *pullback*[1] $f^*(\mathbb{B}^+)$ as the Σ^+-structure with universe A, where the interpretation of a relation symbol R of Σ^+ of arity k is $f^{-1}(R^{\mathbb{B}^+})$, i.e., the inverse image of the interpretation of R in $\mathbb{B}^+$ under the mapping $f : A^k \to B^k$. By definition, $f^*(\mathbb{B}^+)$ is the unique Σ^+-structure on A for which f is a strong homomorphism.

By definition of the structure $\mathbb{J}^C$, there is a distinguished embedding of $\mathbb{J}$ into $\mathbb{J}^C$. Therefore, by composition, any embedding $\pi : \mathbb{A} \to \mathbb{J}$ in $\mathcal{E}$ defines an embedding of $\mathbb{A}$ into $\mathbb{J}^C$, denoted $\hat{\pi} : \mathbb{A} \to \mathbb{J}^C$. Note that for any expansion $\mathbb{J}^+$ of $\mathbb{J}^C$, the pullback $\hat{\pi}^*(\mathbb{J}^+)$ is an expansion of $\mathbb{A}$, which is isomorphic (via $\hat{\pi}$) to an induced substructure of $\mathbb{J}^+$.

▶ **Claim 1.** *There is a coloring $C : \mathcal{E} \to \{L, R\}$ such that, for every expansion $\mathbb{J}^+$ of $\mathbb{J}^C$ over the signature Σ^+, there are two embeddings π and σ in $\mathcal{E}$ such that the pullbacks $\hat{\pi}^*(\mathbb{J}^+)$ and $\hat{\sigma}^*(\mathbb{J}^+)$ are equal, but $C(\pi) \neq C(\sigma)$.*

[1] the word *pullback* has two distinct uses in mathematics – the categorical one, as the dual of a pushout, and another one, used e.g. in differential geometry, as the dual of a pushforward. Our usage is in analogy to the latter.

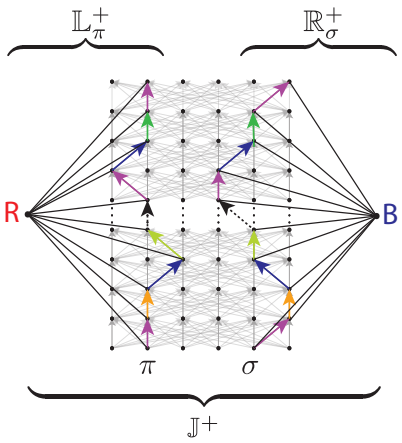
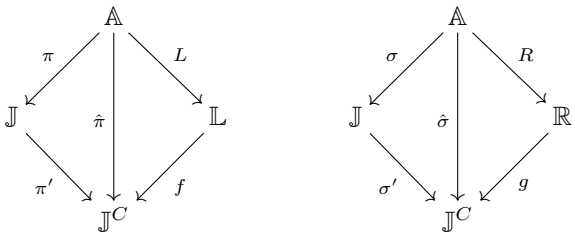


Figure 4 Two embeddings $\pi, \sigma \in \mathcal{E}$ with $C(\pi) = L$ and $C(\sigma) = R$, in the context of the running example. The colored arrows depict various predicates of the stipulated signature Σ^+ extending Σ (in general, they don't need to be binary). The fact that the sequences of colors along π and along σ are the same corresponds to the assumption that the pullbacks $\hat{\pi}^*(\mathbb{J}^+)$ and $\hat{\sigma}^*(\mathbb{J}^+)$ are equal. Therefore, the marked substructures $\mathbb{L}^+_\pi$ and $\mathbb{R}^+_\sigma$ of $\mathbb{J}^+$ (which correspond to $\mathbb{L}^+$ and $\mathbb{R}^+$ in the proof via the mappings f and g) have an isomorphic substructure (isomorphic to $\mathbb{A}^+$ in the proof). An amalgamation in $\mathcal{C}^+$ of $\mathbb{L}^+_\pi$ and $\mathbb{R}^+_\sigma$ along this substructure would yield as a Σ-reduct an amalgamation in $\mathcal{C}$ of $\mathbb{L}$ and $\mathbb{R}$ along $\mathbb{A}$, a contradiction.

We show how the claim yields the lemma. Figure 4 illustrates the proof.

Assume that $\mathcal{C}$ is the class of Σ-reducts of a class of Σ^+-structure $\mathcal{C}^+$. To reach a contradiction, suppose that $\mathcal{C}^+$ is closed under amalgamation. Let C be as in the claim. Since $\mathbb{J}^C$ belongs to $\mathcal{C}$ by the definition of confusion, there exists an expansion $\mathbb{J}^+$ of $\mathbb{J}^C$ in $\mathcal{C}^+$. Let π and σ be as in the conclusion of the claim, and suppose without loss of generality that $C(\pi) = L$ and $C(\sigma) = R$. By the definition of $\mathbb{J}^C$, the embeddings $\pi : \mathbb{A} \to \mathbb{J}$ and $L : \mathbb{A} \to \mathbb{L}$ induce embeddings $\hat{\pi}, \pi', f$, such that the diagram to the left below commutes:



Let $\mathbb{L}^+ = f^*(\mathbb{J}^+)$ be the pullback structure; this structure is an expansion of $\mathbb{L}$. Moreover, $\mathbb{L}^+$ belongs to the class $\mathcal{C}^+$, since it is a pullback under an injective mapping, and hence $\mathbb{L}^+$ is isomorphic to an induced substructure $\mathbb{L}^+_\pi$ of $\mathbb{J}^+$, which is in $\mathcal{C}^+$.

Similarly, the embeddings $\sigma : \mathbb{A} \to \mathbb{J}$ and $R : \mathbb{A} \to \mathbb{R}$ induce embeddings $\hat{\sigma}, \sigma', g : \mathbb{R} \to \mathbb{J}^C$, such that the diagram to the right above commutes. Let $\mathbb{R}^+ = g^*(\mathbb{J}^+)$ be the pullback structure, which is an expansion of $\mathbb{R}$, isomorphic to an induced substructure $\mathbb{R}^+_\sigma$ of $\mathbb{J}^+$, hence belongs to the class $\mathcal{C}^+$.

Let $\mathbb{A}^+$ be the pullback $\hat{\pi}^*(\mathbb{J}^+)$, which, by the claim, is the same as the pullback $\hat{\sigma}^*(\mathbb{J}^+)$. Note that by commutativity of the diagram to the left above, the pullback $\hat{\pi}^*(\mathbb{J}^+)$ is the same as the pullback $L^*(\mathbb{L}^+)$. Similarly, $\hat{\sigma}^*(\mathbb{J}^+)$ is the same as $R^*(\mathbb{R}^+)$. In other words, L is an embedding of $\mathbb{A}^+$ into $\mathbb{L}^+$, and R is an embedding of $\mathbb{A}^+$ into $\mathbb{R}^+$. Since $\mathcal{C}^+$ is closed under

amalgamation, there exists an amalgamation of the diagram $L : \mathbb{A}^+ \to \mathbb{L}^+$ and $R : \mathbb{A}^+ \to \mathbb{R}^+$, which consists of a structure $\mathbb{U}^+$ in $\mathcal{C}^+$ and two embeddings $L' : \mathbb{L}^+ \to \mathbb{U}^+, R' : \mathbb{R}^+ \to \mathbb{U}^+$. Taking Σ-reducts, we obtain an amalgamation in $\mathcal{C}$ of $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$. But L, R were suppose to witness failure of amalgamation in $\mathcal{C}$ – a contradiction proving that $\mathcal{C}^+$ cannot be closed under amalgamation.

Next we show how to prove Claim 1 and hence Lemma 3.4. Call any embedding in $\mathcal{E}$ a *spot*, and any restriction of a spot to an r-element subset of the domain of $\mathbb{A}$ a *partial spot*. For each coloring C of the spots, and each two spots π and σ, define $\pi \approx_C \sigma$ if and only if $C(\pi) = C(\sigma)$. For each coloring D of the partial spots, and each two spots π and σ, define $\pi \sim_D \sigma$ if and only if $D(\pi|_X) = D(\sigma|_X)$ for every r-element subset X of the domain of $\mathbb{A}$. Both are equivalence relations on spots.

▶ **Claim 2.** *There is a coloring C of the spots using two colors, such that for all colorings D of the partial spots using q colors, there is a pair of spots π and σ such that $\pi \sim_D \sigma$ but $\pi \not\approx_C \sigma$.*

We omit the proof of Claim 2, which is by a counting argument relying on the inequality (1).

Finally we use Claim 2 to prove Claim 1. Let C be the coloring of Claim 2 with the two colors interpreted as the embeddings $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$. For each expansion $\mathbb{J}^+$ of $\mathbb{J}^C$, let D be the coloring of partial spots defined as follows. Fix an arbitrary linear ordering of the domain of $\mathbb{A}$, and for each spot π and each r-element subset $X = \{a_1 < \ldots < a_r\}$ of the domain of $\mathbb{A}$, let $D(\pi|_X)$ be the atomic type of the tuple $(\pi(a_1), \ldots, \pi(a_r))$ in $\mathbb{J}^+$. This is a coloring of partial spots using at most q colors. By Claim 2, there is a pair of spots π and σ such that $\pi \sim_D \sigma$ but $\pi \not\approx_C \sigma$. From $\pi \sim_D \sigma$ and the fact that r is at least as large as the maximum arity of any new predicate in Σ^+, it follows that the pullbacks $\hat{\pi}^*(\mathbb{J}^+)$ and $\hat{\sigma}^*(\mathbb{J}^+)$ are equal. On the other hand, from $\pi \not\approx_C \sigma$ we get $C(\pi) \neq C(\sigma)$ by definition. This proves Claim 1 and Lemma 3.4. ◀

4 Classes of consistent structures

In this section we work out the first of our two examples of non-homogenizable classes. We start by defining some basic notions from the theory of constraint satisfaction problems as described, for example, in Chapter 6 of the monograph [12]. Recall that, for a structure $\mathbb{T}$, we write $\mathrm{CSP}(\mathbb{T})$ for the class of all finite structures $\mathbb{I}$ over the same signature as $\mathbb{T}$ for which there is a homomorphism from $\mathbb{I}$ to $\mathbb{T}$. The $\mathbb{I}$'s are called instances, the $\mathbb{T}$'s are called templates.

4.1 Local consistency

Let Σ be a relational signature, let $\mathbb{A}$ and $\mathbb{B}$ be Σ-structures, and let k and l be integers such that $1 \leq k \leq l$. A (k, l)-consistent family on $\mathbb{A}$ and $\mathbb{B}$ is a non-empty family $\mathcal{F}$ of partial homomorphisms from $\mathbb{A}$ to $\mathbb{B}$, such that the following three conditions hold for each f in $\mathcal{F}$:

1. $|\mathrm{Dom}(f)| \leq l$,
2. if X is a subset of $\mathrm{Dom}(f)$, then $f|_X$ is in $\mathcal{F}$,
3. if $|\mathrm{Dom}(f)| \leq k$ and X is a subset of A such that $\mathrm{Dom}(f) \subseteq X$ and $|X| \leq l$, then there exists g in $\mathcal{F}$ such that $\mathrm{Dom}(g) = X$ and $f \subseteq g$.

If there is a (k, l)-consistent family on $\mathbb{A}$ and $\mathbb{B}$, then we say that $\mathbb{A}$ is (k, l)-consistent with respect to $\mathbb{B}$. Note for later use that the class of structures that are (k, l)-consistent with respect to $\mathbb{B}$ is closed under inverse homomorphisms: if there is a homomorphism from $\mathbb{A}'$ to $\mathbb{A}$, and $\mathbb{A}$ is (k, l)-consistent with respect to $\mathbb{B}$, then $\mathbb{A}'$ is also (k, l)-consistent with respect

to $\mathbb{B}$. To see this, it suffices to compose the homomorphism from $\mathbb{A}'$ to $\mathbb{A}$ with each partial homomorphism in the (k,l)-consistent family for $\mathbb{A}$ to get a (k,l)-consistent family for $\mathbb{A}'$.

We describe the special case of $(2,3)$-consistency in terms of a pebble game. The game is played between spoiler and duplicator, each having three pebbles, numbered 1, 2 and 3. Spoiler can place his pebbles on the nodes of $\mathbb{A}$, while duplicator can place his pebbles on the nodes of $\mathbb{B}$. They can also keep the pebbles in their pockets, in which they have all pebbles at the beginning of the game. The game proceeds in rounds as follows. In each round, spoiler places some of the pebbles from his pocket on the nodes of $\mathbb{A}$ and duplicator replies by placing his corresponding pebbles on the nodes of $\mathbb{B}$. If the partial mapping defined by the pebble placement is not a partial homomorphism from $\mathbb{A}$ to $\mathbb{B}$, then duplicator loses. Otherwise, spoiler puts back some of the pebbles into his pocket, and duplicator removes the corresponding pebbles, and the game continues to the next round. It is not hard to see that $\mathbb{A}$ is $(2,3)$-consistent with respect to $\mathbb{B}$ if and only if duplicator can avoid losing forever.

4.2 Systems of linear equations over $\mathbb{F}_2$

We define a finite template $\mathbb{T}_2$ that can be used to represent the solvability of systems of linear equations over the 2-element field. Let us note that our definition of the template $\mathbb{T}_2$ will not be the standard one as it can be found, for example, in the original Feder-Vardi paper [11]. The main difference is that we want to have a signature of smallest possible arity, in this case two. We achieve this by letting $\mathbb{T}_2$ be the natural encoding of the standard template as its *incidence* structure. Concretely, $\mathbb{T}_2$ is defined as follows. Its domain is $D \cup R$, where

$$D = \{0,1\},$$
$$R = \{(x,y,z) \in D^3 : x + y + z = 0 \bmod 2\}.$$

The elements of D are called values, and those of R are called triples. The signature Σ includes three partial functions π_1, π_2, and π_3 that map triples in R to values in D, and four unary relations *value*, *triple*, C_0 and C_1. Formally, in order to have a relational structure, $\mathbb{T}_2$ has binary relations that correspond to the graphs of the partial functions π_1, π_2 and π_3. The interpretations of the symbols in $\mathbb{T}_2$ are as follows:

1. π_1, π_2 and π_3 map (x,y,z) in R to x, y and z, respectively,
2. *value* holds of all elements in D,
3. *triple* holds of all elements in R,
4. C_0 holds of 0 in D, and
5. C_1 holds of 1 in D.

The purpose of *triple* is to encode equations of the type $x + y + z = 0 \bmod 2$, and the purposes of C_0 and C_1 are to encode equations of the type $x = 0$ and $x = 1$, respectively. Note that even though the language does not allow writing more complicated equations, such as $x + y + z = 1 \bmod 2$ or $w + x + y + z = 0 \bmod 2$, such equations can be simulated in the language of $\mathbb{T}_2$ with the help of auxiliary variables.

▶ **Theorem 4.1.** *The class of all finite structures that are $(2,3)$-consistent with respect to $\mathbb{T}_2$ is not homogenizable.*

Proof. In the following, we fix the template $\mathbb{T} = \mathbb{T}_2$, and when we refer to consistency, we mean $(2,3)$-consistency with respect to $\mathbb{T}$. Finite structures on the signature of $\mathbb{T}$ are called instances. Homomorphisms $f : \mathbb{I} \to \mathbb{T}$ from an instance $\mathbb{I}$ to $\mathbb{T}$ are called solutions. By $\mathcal{C}$ denote the class of consistent instances. Observe that, as noted earlier, the class of consistent instances is closed under inverse homomorphisms.

The plan is to apply Theorem 3.2 to $\mathcal{C}$, and for that we need to find a confusing diagram $L : \mathbb{A} \to \mathbb{L}, R : \mathbb{A} \to \mathbb{R}$ with arbitrarily large $\mathbb{A}$.

Let $n \geq 8$ be an exact power of two. Let t be a rooted, ordered complete binary tree with n leaves at depth $\log_2(n)$; in particular, no node at depth 2 is a leaf, and no node at depth $\log_2(n) - 1$ is a root. Let $\mathbb{I}$ be the instance obtained from t, with elements of two types: *nodes*, which correspond to the nodes of t, and *triples*, which correspond to triples (v, v_0, v_1), where v is an internal node in t, and v_0 and v_1 are its left and right sons, respectively. Nodes are labeled by the unary predicate *value* and triples are labeled by the unary predicate *triple*. We say that the triple (v, v_0, v_1) is the triple *below* node v, and is *adjacent* to, or *contains* v, v_0, and v_1. For each such triple, we declare:

$$\begin{aligned} \mathit{father}(v, v_0, v_1) &= \pi_1(v, v_0, v_1) = v, \\ \mathit{left}(v, v_0, v_1) &= \pi_2(v, v_0, v_1) = v_0, \\ \mathit{right}(v, v_0, v_1) &= \pi_3(v, v_0, v_1) = v_1. \end{aligned}$$

We call a structure of this kind simply a tree. Since we will work with Σ-structures that are made of trees, for the sake of intuition from now on we use the names *father*, *left*, and *right* in place of π_1, π_2, and π_3. If i is a value in D, then the *i-marking* of $\mathbb{I}$ is the Σ-structure $m_i(\mathbb{I})$ obtained from $\mathbb{I}$ by marking the root by the predicate C_i. Observe that in any solution $v : m_i(\mathbb{I}) \to \mathbb{T}$ of $m_i(\mathbb{I})$, the sum of the values of the leaves is equal to i modulo 2. Conversely, any mapping from the leaves of $\mathbb{I}$ to $\mathbb{T}$ such that the sum of the values of the leaves is equal to i modulo 2 extends uniquely to a solution $v : m_i(\mathbb{I}) \to \mathbb{T}$.

The structures $\mathbb{L}$ and $\mathbb{R}$ are the markings $m_0(\mathbb{I})$ and $m_1(\mathbb{I})$ of the tree $\mathbb{I}$, respectively. The structure $\mathbb{A}$ is the substructure of $\mathbb{I}$ induced by the leaves of the tree. Note that $\mathbb{A}$ consists of n isolated points, labeled by the unary relation *value*. The unary relations *triple*, C_0 and C_1, as well as the binary relations π_1, π_2, and π_3, are empty in $\mathbb{A}$. Note that $\mathbb{L}$ and $\mathbb{R}$ share $\mathbb{A}$ as an induced substructure. Let $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$ be the corresponding embeddings.

▶ **Lemma 4.2.** *The free amalgam of $\mathbb{L}$ and $\mathbb{R}$ through L and R is inconsistent.*

Proof. When spoiler has only two pebbles on the board, we allow him to perform a move we call a *slide*, in which he moves one pebble from a node v to a triple adjacent to it, or from a triple to a node belonging to this triple. Duplicator has to respond accordingly: if spoiler slides a pebble from a node v to a triple t containing v on the i-th coordinate, then duplicator must move his corresponding pebble from a value v in D to a triple in R containing v on the i-th coordinate. Symmetrically, in the case when spoiler slides his pebble from a triple to the node in the i-th coordinate, duplicator must move his corresponding pebble from the corresponding triple to the value on its i-th coordinate. The slide moves can be simulated in the original game, using a third pebble.

Denote the two (overlapping) trees $\mathbb{I}_L$ and $\mathbb{I}_R$, respectively; they have common leaves in the free amalgam $\mathbb{L} \cup_{\mathbb{A}} \mathbb{R}$. Here is the strategy for spoiler; it consists of several steps. In the beginning of the k-th step, spoiler has two pebbles placed on corresponding nodes a and b of $\mathbb{I}_L$ and $\mathbb{I}_R$, at depth $k-1$ of the tree. In particular, in the beginning of the first step, two pebbles are placed on the roots of $\mathbb{I}_L$ and $\mathbb{I}_R$, respectively. For a node v on which spoiler has his pebble, denote by $r(v)$ the value of the corresponding pebble of duplicator. The invariant is that $r(a) \neq r(b)$. This invariant is clearly satisfied in the beginning of the first step, since $\mathbb{I}_L$ has its root labeled with C_0 and $\mathbb{I}_R$ has its root labeled with C_1.

In the k-th step, spoiler slides his pebble from node a to the triple a' below a in $\mathbb{I}_L$, and then slides his pebble from node b to the triple b' below b in $\mathbb{I}_R$. Duplicator's responses have to satisfy $r(\mathit{father}(a')) = r(a)$ and $r(\mathit{father}(b')) = r(b)$. In particular,

$r(\mathit{father}(a')) \neq r(\mathit{father}(b'))$, by the invariant. It follows that $\mathit{left}(r(a')) + \mathit{right}(r(a')) \neq \mathit{left}(r(b')) + \mathit{right}(r(b'))$, so either $\mathit{left}(r(a')) \neq \mathit{left}(r(b'))$ or $\mathit{right}(r(a')) \neq \mathit{right}(r(b'))$ (or both). Since the cases are symmetric, suppose without loss of generality that the first case occurs. Then spoiler slides the pebble from a' to $\mathit{left}(a')$ and then slides the pebble from b' to $\mathit{left}(b')$, and continues the game from these two nodes playing the role of a and b. The invariant is satisfied.

Since in each step the depth of a increases by 1, at some point, a must be a leaf of $\mathbb{I}_L$, and b is the corresponding leaf in $\mathbb{I}_R$. But then a and b are the same element in $\mathbb{L} \cup_{\mathbb{A}} \mathbb{R}$, and by the invariant $r(a) \neq r(b)$. In other words, spoiler has two pebbles placed at the same node of $\mathbb{L} \cup_{\mathbb{A}} \mathbb{R}$, but the corresponding pebbles of duplicator are not placed on the same element of $\mathbb{T}$. So duplicator loses. ◀

▶ **Lemma 4.3.** *Every amalgam of $\mathbb{L}$ and $\mathbb{R}$ through L and R is inconsistent.*

Proof. This follows at once from the previous lemma and the fact that $\mathcal{C}$ is closed under inverse homomorphisms. Indeed, the free amalgam $\mathbb{L} \cup_{\mathbb{A}} \mathbb{R}$ through L and R maps homomorphically to any amalgam of $\mathbb{L}$ and $\mathbb{R}$ through L and R. ◀

Let m be a natural number, and let $\mathbb{J} = \mathbb{A} \otimes m$, and let $\mathcal{E} = \mathcal{E}_{\mathbb{A},m}$.

The proof of the following lemma is more involved, but is not very insightful, and is omitted due to lack of space.

▶ **Lemma 4.4.** *For every coloring $C : \mathcal{E} \to \{L, R\}$, the structure $\mathbb{J}^C$ is consistent.*

Lemma 4.3 and Lemma 4.4 show that the diagram $L : \mathbb{A} \to \mathbb{L}, R : \mathbb{A} \to \mathbb{R}$ is confusing for the class of consistent structures. Since $\mathbb{A}$ can be taken arbitrarily large, Theorem 4.1 follows immediately from Theorem 3.2. ◀

4.3 Other finite Abelian groups

The template $\mathbb{T}_2$ for systems of equations over the 2-element field can be generalized to all finite Abelian groups. Let G be a finite Abelian group; we write $+$ for the group operation and 0 for its neutral element. Let $\mathbb{T}_G$ be the structure with domain $D \cup R$, where

$$D = G,$$
$$R = \{(x, y, z) \in D^3 : x + y + z = 0\}.$$

The elements of D are called values, and those of R are called triples. As in $\mathbb{T}_2$, the signature of $\mathbb{T}_G$ has three binary relations π_1, π_2, and π_3, two unary relations *value* and *triple*, and one unary relation C_a for each value a in D. The interpretations of all relation symbols are as in $\mathbb{T}$; in particular, the unary relation symbol C_a is interpreted by the singleton set $\{a\}$. It is straightforward to check that $\mathbb{T}_G$ can be used to encode arbitrary systems of equations over G. As in the 2-element field case, equations more complex than the basic $x + y + z = 0$ or $x = a$ can be encoded with the help of auxiliary variables.

▶ **Theorem 4.5.** *If G is a finite Abelian group with at least two elements, then the class of all finite structures that are $(2, 3)$-consistent with respect to $\mathbb{T}_G$ is not homogenizable.*

Proof. The proof of Theorem 4.1 does not rely in any way on the fact that the group is addition mod 2, except for it being Abelian and having at least two different values in it. ◀

It is known that, for any non-trivial finite Abelian group, the constraint satisfaction problem of the template $\mathbb{T}_G$ has *unbounded width*, i.e. for every two natural numbers k and l there exist instances $\mathbb{I}$ that do not have homomorphisms to $\mathbb{T}_G$, but are nonetheless (k,l)-consistent with respect to $\mathbb{T}_G$. We also say that $\mathbb{T}_G$ does not have (k,l)-width for any k and l. This was proved by Feder and Vardi [11] for the standard template for linear equations mod 2, and later alternative proofs generalize quite well to the template $\mathbb{T}_G$ (see, for instance, [1]). Moreover, the solution to the Bounded-Width Conjecture of Barto and Kozik [4] implies that all cases of templates of unbounded width are explained by the unbounded width of some $\mathbb{T}_G$. Technically:

▶ **Theorem 4.6** ([4], see also Theorem 4.1 in [3]). *Let $\mathbb{T}$ be a core finite relational structure. If $\mathbb{T}$ does not have bounded width, then $\mathbb{T}$ pp-interprets $\mathbb{T}_G$ for some non-trivial Abelian group G. Moreover, if the signature of $\mathbb{T}$ has maximum arity at most two, then the conclusion holds even if $\mathbb{T}$ does not have $(2,3)$-width.*

Thus, the templates $\mathbb{T}_G$ are in a strict formal sense the canonical templates of unbounded width. Theorem 4.5 states that, for all such templates, their class of locally consistent instances is non-homogenizable. Interestingly, the converse to this is also true in a strong sense: for all templates that *do have* bounded width, their class of locally consistent instances *is* homogenizable. This follows quite directly from the fact that, for every finite template $\mathbb{T}$, the class of instances $\mathbb{I}$ that have a homomorphism to $\mathbb{T}$ is homogenized by expanding them by all their homomorphisms to $\mathbb{T}$. When these two observations are put together, we get that, up to the relation of pp-interpretability between templates, which is known to preserve the property of having bounded width, the templates that have bounded width are distinguished from those that do not by the fact that their classes of locally consistent instances are homogenizable. It seems plausible that our Theorem 4.5 could be adapted to show that *all* templates of unbounded width give themselves a non-homogenizable class of locally-consistent instances, without the need to resort to pp-interpretability, but this remains open.

5 Classes defined by forbidden homomorphisms

The positive result of Hubička and Nešetřil [14] shows that if $\mathcal{F}$ is an HN-regular class of finite connected structures, then the class $\mathrm{Forb}_h(\mathcal{F})$ is a reduct of an amalgamation class. HN-regularity is a notion reminiscent of the notion of regularity of word languages or of tree languages. Indeed, in the case of structures of treewidth one, HN-regularity and regularity in the sense of tree automata both correspond to MSO-definability. In this section we give an example of a non-homogenizable class of finite structures that is of the form $\mathrm{Forb}_h(\mathcal{G})$, where $\mathcal{G}$ is an MSO-definable class of connected finite structures of treewidth two. It follows from the result of Hubička and Nešetřil that this is optimal. Recall that the treewidth of a finite structure is defined as the treewidth of its Gaifman graph, and pathwidth is a restriction of treewidth (for definitions see, for example, [8]).

5.1 Pathwidth three

Consider the class $\mathcal{F}$ from the running example in Section 3, shown not to be a reduct of any amalgamation class in Example 3.3. The class $\mathcal{F}$ can be defined by an MSO sentence, which expresses that there are exactly four colored points, which are colored R, B, S, and T, respectively, and the rest of points form a directed simple $\vec{E}$-path from S to T with all vertices along the path connected by an undirected E-edge to both R and B. Moreover, each

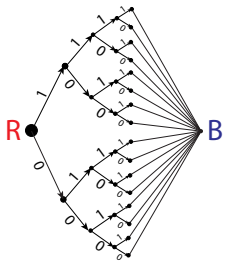


Figure 5 Forbidden structure $\mathbb{G}$, with 16 leaves.

structure $\mathbb{F}$ in $\mathcal{F}$ is connected and has pathwidth three: take a standard path-decomposition of the $\vec{E}$-path with bags of size 2 and add both red and blue vertices to each bag. This gives a path-decomposition with bags of size 4, so its pathwidth is 3 (thanks to the -1 in the definition of treewidth/pathwidth).

Now we show how to modify the class $\mathcal{F}$ to obtain a class of structures of treewidth two.

5.2 Treewidth two

Consider a rooted, directed binary tree, in which every node is either a leaf, or an inner node with two sons, in which case it has a directed $\vec{E}_0$-edge to its left son and a directed $\vec{E}_1$-edge to its right son. Color its root red, by labeling it with the unary predicate R, and create an extra blue vertex (with unary predicate B), connected to all the leaves of the tree by an undirected E-edge. An example of such a structure, obtained from a full binary tree of depth 4, is depicted in Figure 5. Let $\mathcal{G}$ denote the class of all structures obtained in this way.

The signature Σ of these structures consists of three binary predicates $E, \vec{E}_0, \vec{E}_1$, and two unary predicates R and B, each appearing in the structure exactly once as indicated. It is straightforward to check that $\mathcal{G}$ is MSO-definable on the class of all finite structures. Moreover, each structure $\mathbb{G}$ in $\mathcal{G}$ is connected and has treewidth two: just take a tree-decomposition of the binary tree and add the blue point to all its bags.

▶ **Claim 3.** *The class $\mathcal{G}$ forms an antichain in the homomorphism preorder.*

Proof. Suppose that $h : \mathbb{G}_1 \to \mathbb{G}_2$ is a homomorphism of two structures in $\mathcal{G}$. Then h must map the root of $\mathbb{G}_1$ to the root of $\mathbb{G}_2$ (since only the root is colored red), and must map the leaves of $\mathbb{G}_1$ to the leaves of $\mathbb{G}_2$ (since only the leaves are adjacent to a blue node). Finally, a vertex v in $\mathbb{G}_1$ reached from the root by a path with labels $i_1 i_2 \dots i_k \in \{0,1\}^*$ must be mapped to the unique vertex w of $\mathbb{G}_2$ reached from the root by the path obtained by reading the same labels. Hence, the mapping f is injective. Since no inner node of the tree can be mapped to a leaf, f must also be surjective. It follows that h is an isomorphism. ◀

▶ **Proposition 5.1.** *The class* $\mathrm{Forb}_h(\mathcal{G})$ *is not homogenizable.*

Proof. We apply Theorem 3.2. To this end, choose an arbitrary structure $\mathbb{G} \in \mathcal{G}$, and consider the diagram $L : \mathbb{A} \to \mathbb{L}, R : \mathbb{A} \to \mathbb{R}$ defined as follows. $\mathbb{L}$ is the left part of the structure $\mathbb{G}$, obtained by removing the blue vertex (labeled B), $\mathbb{R}$ is the right part of the structure $\mathbb{G}$, obtained by keeping the blue vertex and the nodes adjacent to it, $\mathbb{A}$ is the intersection of $\mathbb{L}$ and $\mathbb{R}$, i.e., the substructure of $\mathbb{G}$ induced by the leaves of the underlying binary tree. Let $L : \mathbb{A} \to \mathbb{L}$ and $R : \mathbb{A} \to \mathbb{R}$ be the two inclusions. It is clear that every amalgamation of L, R must contain a homomorphic image of $\mathbb{G}$, so L, R witnesses failure

of amalgamation of $\mathrm{Forb}_h(\mathcal{G})$. Let m be an arbitrary number, $\mathbb{J} = \mathbb{A} \otimes m$, $\mathcal{E} = \mathcal{E}_{\mathbb{A},m}$, and $C : \mathcal{E} \to \{L, R\}$ be any coloring.

▶ **Claim 4.** *The structure $\mathbb{J}^C$ does not contain a homomorphic image of any structure in $\mathcal{G}$.*

The proof uses similar ideas as in the proof of Claim 3, and is omitted.

Hence, the diagram L, R is confusing for $\mathcal{C}$. Since $\mathbb{G}$ can be chosen so that $\mathbb{A}$ is arbitrarily large, the conclusion follows from Theorem 3.2. ◀

5.3 Optimality

We argued already that the classes $\mathcal{F}$ from Example 3.1 and $\mathcal{G}$ from Section 5.2 are MSO-definable. Therefore, the set of colored paths that represent the path-decompositions of the structures in $\mathcal{F}$ is regular in the automata-theoretic sense, and the set of colored trees that represent the tree-decompositions of the structures in $\mathcal{G}$ is regular in the tree-automata-theoretic sense (see [8]). It is interesting to check why $\mathcal{F}$ and $\mathcal{G}$ are *not* regular classes of structures in the sense of Definition 2.3 of Hubička-Nešetřil [14]. By Example 3.3 and Proposition 5.1 we know that $\mathcal{F}$ and $\mathcal{G}$ cannot be HN-regular as otherwise $\mathrm{Forb}_h(\mathcal{F})$ and $\mathrm{Forb}_h(\mathcal{G})$ would be homogenizable by Theorem 3.1 in [14].

In order to check that a class is not HN-regular it suffices to identify minimal g-separating g-cuts of unbounded sizes in its structures. For $\mathcal{F}$, note that the set of all vertices in the $\vec{E}$-path is a minimal g-separating g-cut in $\mathbb{F}_k$, and its size is k and hence unbounded. For $\mathcal{G}$, the set of all leaves in the binary tree in any structure $\mathbb{G}$ in $\mathcal{G}$ is a minimal g-separating g-cut, and its size is also unbounded since all trees are represented in $\mathcal{G}$.

We note that every MSO-definable class of finite connected structures of treewidth one *is* HN-regular. This follows from the fact noted earlier that, for colored trees, HN-regularity, tree-automata regularity, and MSO-definability are equivalent. In particular, by Theorem 3.1 in [14], every class of the form $\mathrm{Forb}_h(\mathcal{F})$, where $\mathcal{F}$ is an MSO-definable class of connected finite structures of treewidth at most one, is homogenizable.

Conclusion

We study homogenizability – a combinatorial notion useful in computer science (see e.g. [5],[7]). Our main contribution is a necessary condition for homogenizability of a class of finite structures. We apply it to prove nonhomogenizability of a class related to constraint satisfaction problems, consisting of locally consistent structures with respect to the template encoding linear equations over a finite abelian group, and an MSO-definable class of structures of treewidth two, which is tight by the positive result of [14].

Acknowledgment. We thank the anonymous reviewers for the useful remarks.

References

1 A. Atserias, A. A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.*, 410(18):1666–1683, 2009.

2 A. Atserias and M. Weyer. Decidable relationships between consistency notions for constraint satisfaction problems. In *Proceedings of the 23rd CSL International Conference and 18th EACSL Annual Conference on Computer Science Logic*, CSL'09/EACSL'09, pages 102–116, Berlin, Heidelberg, 2009. Springer-Verlag.

3 L. Barto. The constraint satisfaction problem and universal algebra. *The Bulletin of Symbolic Logic*, 21:319–337, 9 2015.

4 L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, January 2014.

5 M. Bodirsky. *Complexity of Constraints: An Overview of Current Research Themes*, chapter Constraint Satisfaction Problems with Infinite Templates, pages 196–228. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

6 M. Bojańczyk, B. Klin, and S. Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3:4):paper 4, 2014.

7 M. Bojańczyk, L. Segoufin, and S. Toruńczyk. Verification of database-driven systems via amalgamation. In Richard Hull and Wenfei Fan, editors, *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA – June 22-27, 2013*, pages 63–74. ACM, 2013. doi:10.1145/2463664.2465228.

8 B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

9 J. Covington. Homogenizable relational structures. *Illinois J. Math.*, 34(4):731–743, 12 1990. URL: http://projecteuclid.org/euclid.ijm/1255988065.

10 P. L. Erdös, C. Tardif, and G. Tardos. On infinite-finite duality pairs of directed graphs. *Order*, 30(3):807–819, 2013.

11 T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

12 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007.

13 W. Hodges. *A Shorter Model Theory*. Cambridge University Press, New York, NY, USA, 1997.

14 J. Hubička and J. Nešetřil. Universal structures with forbidden homomorphisms. In *Logic Without Borders: Essays on Set Theory, Model Theory, Philosophical Logic and Philosophy of Mathematics*, pages 241–264. De Gruyter, 2015.

15 Ph. G. Kolaitis, H. J. Proemel, and B. L. Rothschild. K_{l+1}-Free Graphs: Asymptotic Structure and a $0-1$ Law. *Transactions of the American Mathematical Society*, 303(2):637–671, 1987.

16 D. Macpherson. A survey of homogeneous structures. *Discrete Mathematics*, 311(15):1599–1634, 2011. Infinite Graphs: Introductions, Connections, Surveys.

17 J. Nešetřil. Ramsey theory. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics (Vol. 2)*, pages 1331–1403. MIT Press, Cambridge, MA, USA, 1995.

Context-Free Graph Properties via Definable Decompositions

Michael Elberfeld

RWTH Aachen University, Aachen, Germany
elberfeld@informatik.rwth-aachen.de

Abstract

Monadic-second order logic (MSO-logic) is successfully applied in both language theory and algorithm design. In the former, properties definable by MSO-formulas are exactly the regular properties on many structures like, most prominently, strings. In the latter, solving a problem for structures of bounded tree width is routinely done by defining it in terms of an MSO-formula and applying general formula-evaluation procedures like Courcelle's. The present paper furthers the study of second-order logics with close connections to language theory and algorithm design beyond MSO-logic.

We introduce a logic that allows to expand a given structure with an existentially quantified tree decomposition of bounded width and test an MSO-definable property for the resulting expanded structure. It is proposed as a candidate for capturing the notion of "context-free graph properties" since it corresponds to the context-free languages on strings, has the same closure properties, and an alternative definition similar to the one of Chomsky and Schützenberger for context-free languages. Besides studying its language-theoretic aspects, we consider its expressive power as well as the algorithmics of its satisfiability and evaluation problems.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases finite model theory, monadic second-order logic, tree decomposition, context-free languages, expressive power

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.17

1 Introduction

The properties that are definable by formulas from monadic second-order logic (MSO-logic) correspond to the notion of regularity for many classes of structures. Most prominently, the MSO-definable properties of strings correspond to the regular languages [3, 8, 19] and this also holds for regular languages of trees [18, 6]. Moreover, a recent work [2] generalizes this connection to the MSO-definable properties of every class of structures with bounded tree width (that means, a class of structures that have tree decompositions whose width is bounded by a class-dependent constant).

Besides its application to study regular languages of various classes of tree-width-bounded structures, MSO-logic is frequently used to design algorithms for solving problems on these structures. Every MSO-definable property can be decided in polynomial time on every class of structures of bounded tree width, even if we are looking out for an algorithm that runs in linear time [4] or has a logarithmic memory footprint [7]. The usefulness of these results lies in the fact that, in order to prove that a certain concrete problem is decidable in, say, linear time on a class of structures of bounded tree width, we only need to show that it is MSO-definable. That means, writing down a defining MSO-formula for it. Then the linear time bound for solving the problem follows from the general result of [4].

© Michael Elberfeld;
licensed under Creative Commons License CC-BY

Besides MSO-definable problems, there are many problems of both practical and theoretical interest that are not MSO-definable. In order to define them, we need second-order formulas, which have the ability to establish new relations. Prominent examples of such properties are the context-free languages of strings. Since there are context-free languages that are not regular, they are not definable in terms of MSO-formulas. Lautemann, Schwentick, and Thérien [14] addressed this issue by presenting a logic for the context-free languages of strings. It extends MSO-logic with the ability to establish a binary relation, called a *matching* relation, that puts a string of well-formed nested brackets on top of a given string. While this logic generalizes the connection between formal languages and predicate logics from regular languages to context-free languages, it does not apply to more general structures like trees and graphs of bounded tree width. In particular, it does not provide us with a logic that helps to further the application of logics as a descriptive tool in algorithm design. Motivated by the comment of Lautemann et al. that "We are convinced that a more general study of these logics will prove worthwhile in the context of general finite structures, instead of strings", the present paper remedies this situation by presenting an extension of MSO-logic that is equivalent to the context-free languages of strings, and also generalizes to every class of structures of bounded tree width.

Contributions. We introduce a logic that generalizes MSO on bounded tree width graphs by (1) allowing to existentially guess a tree decomposition of some bounded width w for a given graph, and (2) test an MSO-definable property for a structure that encodes both the given graph and the guessed tree decomposition. That means, the logic still defines a property of graphs, but the defined property is based on cycling through candidate tree decompositions of them. We call the newly proposed logic MSO-*logic with quantified tree decompositions* (DMSO-logic). The main purpose of the present paper is to propose this logic as a candidate for capturing the notion of "context-free properties of graphs" with strong connections to both formal languages and algorithm design.

In order to tighten the connection of DMSO-logic with formal languages, we prove that its definable properties of strings are exactly the context-free languages while already a slight modification of the decompositions that we use leads to a logic that also defines properties that are not context-free. Moreover, we study the closure of DMSO-definable properties under set operations and show that they are similar to the closure properties of the context-free languages. Our logic also allows for an alternative characterization similar to the Theorem of Chomsky and Schützenberger, which characterizes context-free languages in terms of words containing well-nested brackets and a regular property of these words.

For establishing a connection between DMSO-logic and algorithm design, we study the complexity of its satisfiability and formula-evaluation problems. Its satisfiability problem turns out to be decidable. Deciding whether a given structure is a model of a DMSO-formula turns out to be polynomial-time computable for every fixed formula. In order to obtain the polynomial-time upper bound, we need to work with MSO-properties of candidate tree decompositions without constructing them explicitly. We work with logical types and develop an inductive approach that computes the types of substructures and their candidate decompositions. In order to understand the range of applications of our logic, we also study its expressive power. Since MSO-logic on structures of bounded tree width is well studied, we concentrate on the main distinguishing feature of DMSO-logic: the quantified tree decompositions. We study how a varying width of the quantified tree decomposition influences the expressive power of the logic. Again, using logical types to reasoning about candidate decompositions plays an important role for proving the expressivity results.

Organization. After introducing the necessary background on MSO-logic in Section 2, we define DMSO-logic in Section 3. Sections 4, 5, and 6 present results related to the language-theoretic, model-theoretic, and algorithmic aspects of DMSO-logic, respectively. Section 7 concludes with a discussion of the presented results and directions for future work.

2 Background on Monadic Second-Order Logic

A *vocabulary* τ is a finite set of *relational symbols* where an *arity* $\mathrm{ar}(R) \geq 1$ is assigned to each symbol $R \in \tau$. A *structure* A over τ (also called τ-structure) consists of a finite set $U(A)$, its *universe*, and a *relation* $R(A) \subseteq U(A)^{\mathrm{ar}(R)}$ for every $R \in \tau$. The *tuples* of A are all $(a_1, \ldots, a_{\mathrm{ar}(R)}) \in R(A)$ for some $R \in \tau$. The substructure of A that is *induced* by a set of elements $U' \subseteq U(A)$ is denoted by $A[U']$; it has universe U' and consists of all tuples from A whose elements are in U'. We view a *graph* G as a structures over the vocabulary $\tau_{\text{graphs}} := \{E^2\}$; in addition to the above notation we denote its universe also by $V(G)$ and call its elements *vertices*, and call the tuples in $E(G)$ *edges*. A graph is *undirected* if $E(G)$ is symmetric.

To define the *syntax* of *second-order logic* (SO-logic), we use *element variables* (also called *first-order variables*) x_i for $i \in \mathbb{N}$ and *relation variables* (also called *second-order variables*) X_i for $i \in \mathbb{N}$ that have an arity $\mathrm{ar}(X_i) \geq 1$. *Formulas of* SO*-logic* (SO*-formulas*) over a vocabulary τ are inductively defined as follows: *Atomic* formulas are all $x_i = x_j$, $(x_{i_1}, \ldots, x_{i_r}) \in X_j$ with $r = \mathrm{ar}(X_j)$, and $(x_{i_1}, \ldots, x_{i_r}) \in R$ for $R \in \tau$ with $r = \mathrm{ar}(R)$. Given SO-formulas φ_1 and φ_2, *composed* formulas are build from *element quantifiers* via $\exists x_i\,(\varphi_1)$ and *relation quantifiers* via $\exists X_i\,(\varphi_1)$ as well as *Boolean connectives* $\varphi_1 \wedge \varphi_2$ and $\neg\,(\varphi_1)$. The set of *free variables* of an SO-formula φ, denoted by $\mathrm{free}(\varphi)$, contains the variables of φ that are not used as part of a quantification. By renaming a formula's variables, we can always assume $\mathrm{free}(\varphi) = \{x_1, \ldots, x_k, X_1, \ldots, X_\ell\}$ for some $k, \ell \in \mathbb{N}$; we write $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ to indicate that the free variables of φ are exactly x_1 to x_k and X_1 to X_ℓ. An SO*-sentence* is an SO-formula without free variables; we use the term sentence in a similar way for other formulas from other logics as well. Regarding the *semantics* of SO-logic, we write $A \models \varphi(a_1, \ldots, a_k, A_1, \ldots, A_\ell)$ to indicate that an SO-formula $\varphi(x_1, \ldots, x_k, X_1, \ldots, X_\ell)$ evaluates to true for a structure A over the same vocabulary and an *assignment* $(a_1, \ldots, a_k, A_1, \ldots, A_\ell) \in U^k \times \mathrm{pow}(U^{\mathrm{ar}(X_1)}) \times \cdots \times \mathrm{pow}(U^{\mathrm{ar}(X_\ell)})$ to φ's free variables, where $\mathrm{pow}(B)$ denotes the power set of a set B.

Monadic second-order logic (MSO-logic) is defined by taking all SO-formulas without second-order variables (neither free nor bound) of arity 2 and higher along with the semantics of SO-logic; we call the second-order variables just *set variables* when working with MSO-logic. *Modulo-counting monadic second-order logic* (CMSO-logic) is an extension of MSO-logic that also allows to use *modulo-counting atomic formulas* $\mathrm{MOD}_m(X)$ for every $m \in \mathbb{N}$ and set variable X. We write $A \models \mathrm{MOD}_m(A_1)$ for a set $A_1 \subseteq U(A)$ exactly if the number of elements in A_1 is a multiple of m.

A set of τ-structures $\mathcal{P}$ that is closed under taking isomorphic structures is also called a *property* of τ-structures. It is SO*-definable* if there is an SO-formula φ over τ with $A \models \varphi$ for τ-structures $A \in \mathcal{P}$ and $A \not\models \varphi$ for τ-structures $A \notin \mathcal{P}$. Given a second set of structures $\mathcal{C}$, we say that $\mathcal{P}$ is SO*-definable on* $\mathcal{C}$ if $\mathcal{P} \cap \mathcal{C}$ is SO-definable. Note that this definition does not consider membership in $\mathcal{C}$ as a promise, which all given structures meet by assumption. We use the notion of *definable* and *definable on* also for other logics like, for example, MSO-logic and CMSO-logic.

3 Monadic Second-Order Logic with Quantified Tree Decompositions

The present section introduces a logic that we propose as an accessible framework for capturing the notion of context-free properties of graphs and, more general, structures. It is based on guessing a constant-width tree decomposition for a given structure and checking a property for the decomposition and the structure that is MSO-definable. In order to introduce the logic, we review the notion of tree decompositions and how they can be encoded as part of structures in Section 3.1. Based on this concept, we define our logic in Section 3.2.

3.1 Structures Expanded by Tree Decompositions

A *directed tree* T is a tree whose edges are oriented away from a unique *root* $\mathrm{r}(T) \in V(T)$; we call is vertices *nodes* and edges *arcs*. A *tree decomposition* $D = (T, \beta)$ of a structure A is a directed tree T together with a labeling function $\beta\colon V(T) \to \mathrm{pow}(U(A))$ satisfying the below connectedness, covering, and strictness conditions. For every $t \in V(T)$, the set $\beta(t)$ is the *bag* of t. The *width* of D is $\max_{t \in V(T)} |\beta(t)| - 1$.

- *(Connectedness)* For every element $a \in U(A)$, the induced substructure $T\big[\{t \in V(T) \mid a \in \beta(t)\}\big]$ is a nonempty directed tree.
- *(Covering)* For every tuple $(a_1, \dots, a_r)$ of A, there is a $t \in V(T)$ with $\{a_1, \dots, a_r\} \subseteq \beta(t)$.

In order to state the strictness condition, we introduce some terminology partly adapted from [11]. The root of $D = (T, \beta)$ is $\mathrm{r}(D) := \mathrm{r}(T)$. The *separator* of t is $\sigma(t) := \beta(u) \cap \beta(t)$ if it has a parent u and $\sigma(t) := \emptyset$ if $t = \mathrm{r}(T)$. For every $t \in V(T)$, T_t is the subtree of T rooted at t. Then $\gamma(t)$, called the *cone* of t, is the union of all bags $\beta(u)$ for $u \in V(T_t)$. The *component* of t is $\alpha(t) := \gamma(t) \setminus \sigma(t)$. Besides the two conditions above, we require the following third condition that is commonly met by the tree decompositions used in the literature, but normally not made explicit as part of the definition.

- *(Strictness)* For every $(u, t) \in E(T)$, we have $\gamma(u) \supsetneq \gamma(t)$ or $\alpha(u) \supsetneq \alpha(t)$.

The *Gaifman graph* $G(A)$ of a structure A has vertex set $V(G(A)) := U(A)$ and there is an edge $(u, v) \in E(G(A))$ exactly if u and v are part of a common tuple t of A; thus, $G(A)$ is undirected. Instead of directly working with tree decompositions for structures, we sometimes use the following folklore fact (see [10] for a formal proof) to consider tree decompositions for their Gaifman graphs.

▶ **Fact 1.** *Every tree decomposition of a structure A is also a tree decomposition of its Gaifman graph $G(A)$, and vice versa.*

Let A be a structure over some vocabulary τ and $D = (T, \beta)$ a tree decomposition for it. In order to have a single structure that encodes both A and D, we define the *expansion* (A, D) of A by D as follows: Its universe is the disjoint union of $U(A)$ and $V(T)$. It contains all relations of A along with the unary relation $\textsc{nodes} := V(T)$, which is used to distinguish between nodes from the tree decomposition and elements from the structure, the binary relation $\textsc{arcs} := E(T)$, to encode the directed tree underlying the tree decomposition, and $\textsc{bags} := \{(t, a) \in V(T) \times U(A) \mid a \in \beta(t)\}$, to encode the assignment of elements to the bags. Thus, (A, D) is a structure over the vocabulary $\tau^\star := \tau \cup \{\textsc{nodes}, \textsc{arcs}, \textsc{bags}\}$ where we assume that the relation symbols NODES, ARCS, and BAGS are not part of τ. If we move from an expansion (A, D) back to A, we speak of *reducing* (A, D). Courcelle and Engelfriet [5, Example 5.2] discuss other ways of how to encode a graph along with a tree decomposition as a single relational structure. The definition in the present paper follows the work of Adler et al. [1].

The tree width $\mathrm{tw}(A)$ of a structure A is the minimum width among all its tree decompositions. The following lemma implies that the tree width of a structure only grows by a constant additive factor if we expand it with a tree decomposition of minimum possible width. The lemma without a proof can also be found in [1].

▶ **Lemma 2.** *Let A be a structure with a tree decomposition $D = (T, \beta)$ of width w. Then the tree width of (A, D) is at most $w + 2$.*

Proof. We show how to build a width-$(w + 2)$ tree decomposition $D' = (T', \beta')$ for (A, D), which proves the lemma. As the underlying tree T', we use a copy of T and rename each node $t \in V(T)$ into t'. We set $\beta'(t') := \{u, t\} \cup \beta(t)$ if t has a parent u in T and $\beta'(t') := \{t\} \cup \beta(t)$ if $t = \mathrm{r}(T)$. Since D is already a tree decomposition for A, this construction satisfies the connectedness condition for all elements from $U(A)$ and the cover condition for all tuples from A. A node u from $V(T)$ only appears in the bag of u' and the bag of each t' where t is a child of u in T. Thus, the connectedness condition holds for the elements from $V(T)$. Moreover, the construction puts every edge $(u, t) \in E(T)$ into the bag $\beta'(t')$. Strictness is inherited from D and the width bound $w + 2$ follows from the construction. ◀

3.2 Quantified Tree Decompositions

For the definition of the logic, we use tree decompositions $D = (T, \beta)$ for structures A that are normalized in the sense of satisfying the following condition.

- (Normal) For every $t \in V(T)$, $G[\alpha(t)]$ is connected, where $G = G(A)$.

This condition, which is commonly met by tree decompositions that are constructed via recursive separator-based approaches, is a necessary ingredient for proving the equivalence to the context-free languages of our logic on strings. This can be seen by comparing the tree decompositions used in Theorem 6, which are normal, with the tree decompositions used in Lemma 10, which are not normal in general.

Compared to MSO-logic, the following logic allows to establish new relations beyond just sets, but the relations encode a tree decomposition of bounded width. Definition 4 uses the MSO-sentences from the following proposition, which can be written down by formulating the above conditions for tree decompositions in terms of subformulas.

▶ **Proposition 3.** *For every vocabulary τ and $w \in \mathbb{N}$, there is an* MSO*-sentence $\varphi_{\text{tw-}w\text{-dec}}$ over $\tau^\star$ that defines the τ-structures that are expanded by normal tree decompositions of width at most w.*

▶ **Definition 4.** We define the syntax and semantics of MSO*-logic with quantified tree decompositions* (DMSO-logic) as follows:

- (Syntax) A DMSO-formula over some vocabulary τ has the form $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ where $\varphi_{\text{tw-}w\text{-dec}}$ is from Proposition 3 and φ is an MSO-sentence over $\tau^\star$.
- (Semantics) For a τ-structure A and a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$, we write $A \models \psi$ exactly if there is a tree decomposition D with $(A, D) \models \varphi_{\text{tw-}w\text{-dec}} \wedge \varphi$.

Similar to how CMSO-logic extends MSO-logic, we extend DMSO-logic to a modulo-counting variant by allowing φ to be a CMSO-sentence instead of just an MSO-sentence. The resulting logic is called CMSO*-logic with quantified tree decompositions* (DCMSO-logic).

While DMSO-logic implicitly guesses a tree decomposition, which includes establishing new elements, it can, equivalently, be defined as a syntactic fragment of SO-logic. In this case, we use higher-ary tuples to encode elements of the tree decomposition. In order to have

a clean presentation, we use the definition given above, but keep in mind that DMSO-logic can be defined as a syntactic fragment of SO-logic.

The property defined by the MSO-subformula φ in Definition 4 can, possibly, depend on the shape of the tree decomposition that is guessed by the decomposition quantifier. A variant of this definition would be to allow an existentially guessed tree decomposition, but restrict the property of φ to be invariant with respect to the chosen tree decomposition. In this case, the guessed tree decomposition supports the formula φ in defining a property of the given structure rather than φ defining a property of the structure and the tree decomposition. Interestingly, this invariant use of a width-bounded tree decomposition corresponds to the notion of recognizable properties of graphs of bounded tree width as studied in [2].

4 Language-Theoretic Aspects

The present section shows that the DMSO-definable properties of strings are exactly the context-free languages (Section 4.1). Moreover, we show that modifying DMSO-logic in terms of using decompositions that are not normal leads to a more expressive logic (Section 4.2).

4.1 DMSO-Definability on Strings Equals the Context-Free Languages

In order to move back and forth between formal languages and logic, we encode strings by structures that have the shape of a path with unary relations to encode symbols. For an alphabet $\Sigma = \{S_1, \dots, S_m\}$, we consider the vocabulary $\tau_\Sigma := \{S_1^1, \dots, S_m^1\}$. A Σ-*path* is a structure P over the vocabulary $\tau_{\Sigma\text{-paths}} := \text{SUCC}^2 \cup \tau_\Sigma$ where $\text{SUCC}(P)$ is the edge relation of a directed path and the unary relations $S_1(P)$ to $S_m(P)$ partition $U(P)$. Strings over an alphabet Σ translate into Σ-paths and back if we use $\text{SUCC}(P)$ to encode the concatenation and the $S_i(P)$ to encode symbols. Given a language $L \subseteq \Sigma^*$ for some alphabet Σ, we denote by $\mathcal{P}(L)$ the set of Σ-paths that encode strings from L.

Based on this notation, we are able to state the equivalence of the regular languages and the MSO-definable string properties of Büchi, Elgot, and Trakhtenbrot [3, 8, 19].

▶ **Fact 5.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is regular if, and only if, $\mathcal{P}(L)$ is* MSO*-definable.*

Similar to this fact, during the course of the present section, we prove the following connection between the DMSO-definable properties of strings and the context-free languages.

▶ **Theorem 6.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is context-free if, and only if, $\mathcal{P}(L)$ is* DMSO*-definable.*

The theorem follows from Lemmas 8 and 9, which show how to move from context-free languages to DMSO-definable string properties and back, respectively. Moreover, the lemmas imply that, in order to capture the context-free languages, we only need tree decompositions of width at most 3. By Lemma 9, we do not leave the context-free languages if we use tree decompositions with a width bound that is larger than 3, but we can always go down to width 3 by applying Lemma 8 to the result of Lemma 9.

Besides using a context-free grammar or a pushdown automaton, a context-free language can be defined as the set of strings that are the yields of trees of a regular tree language; meaning that a string is generated by taking a binary tree with distinguished left and right children and reading the symbols at the leaves from left to right with respect to the ordering induced by the inner nodes of the tree. Since regular tree languages are exactly the MSO-definable properties of trees, this gives rise to a descriptive characterization of the

context-free languages in terms of trees and MSO-logic. We review this connection between context-free languages and MSO-definable trees in order to prove Lemmas 8 and 9.

A Σ-*tree* is a structure T over the vocabulary $\tau_{\Sigma\text{-trees}} := \{\text{LEFT}^2, \text{RIGHT}^2\} \cup \tau_\Sigma$ where relations $\text{LEFT}(T)$ and $\text{RIGHT}(T)$ encode a directed binary tree whose edges are partitioned into *left successors* $\text{LEFT}(T)$ and *right successors* $\text{RIGHT}(T)$. The unary relations partition the nodes of the tree (in fact, for the application to a characterization of the context-free languages, we only need that they partition the leaves of the tree). A Σ-tree *yields* a Σ-path as follows: Starting at T's root $\text{r}(T)$, we traverse the tree based on the depth-first search that walks to left successors with higher priority and to right successors with lower priority. This walk induces a total ordering on the leaves of the tree. In turn, it gives rise to a Σ-path P whose nodes are the leaves of T; this path is the *yield* of T.

We get the following fact from combining [16] with [18, 6] (see also its application in [14]).

▶ **Fact 7.** *Let $L \subseteq \Sigma^*$ for an alphabet Σ. Then L is context-free if, and only if, there is an* MSO*-definable property of Σ-trees $\mathcal{T}$ that yield exactly the Σ-paths $\mathcal{P}(L)$.*

▶ **Lemma 8.** *For every* MSO*-sentence φ over $\tau_{\Sigma\text{-trees}}$, there is an* MSO*-sentence $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$ satisfying the following for every Σ-path P: there is a Σ-tree T that yields P with $T \models \varphi$ if, and only if, there is a width-3 normal tree decomposition D for P with $(P, D) \models \varphi^\star$.*

Proof. For the proof we use a transformation from Σ-trees that yield paths to tree decompositions for paths. Based on this transformation, we turn a formula φ over $\tau_{\Sigma\text{-trees}}$ into a formula $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$ that meets the requirements of the lemma.

We start to describe how a Σ-tree T that yields a path P is turned into a tree decomposition $D = (T', \beta)$ for P. The tree underlying the decomposition is the tree T without distinguishing left and right successors where we rename every node $t \in V(T)$ into $t' \in V(T')$. That means $V(T') := V(T)$ and $E(T') := \text{LEFT}(T) \cup \text{RIGHT}(T)$ up to renaming nodes. We view the nodes of T' as being partitioned into the set of leaves, which are exactly the elements of P, and the inner nodes. The bags $\beta\colon V(T') \to \text{pow}(U(P))$ are defined as follows: For every $t' \in V(T')$, let P_t be the path that is the yield of the subtree T_t rooted at t. If t' is a leaf, then set $\beta(t') := U(P_t) = \{t\}$. If t' is an inner node, let l' and r' be its left child and right child, respectively, in T. Then $\beta(t')$ contains 4 elements of P_t: the left-most and right-most nodes of both P_l and P_r. Then $D = (T', \beta)$ is a tree decomposition for P due to the following reasons: Every unary tuple of P is already covered by the bags of the leaves. For a binary tuple $(p_1, p_2) \in \text{SUCC}(P)$, let l be the highest node in T, such that the right-most node of P_l is p_1, and let r be the highest node in the tree, such that the left-most node of P_r is p_2. The nodes l and r are children of a common parent node t and $\{p_1, p_2\} \subseteq \beta(t')$ holds. The connectedness condition is satisfied due to the following reason: A node of P is part of a leaf bag, stays in the bags above it while it is the left-most or right-most node of the path that is the yield of a subtree, stays in the bags while the two subtrees are merged, and is deleted from the bag afterwards. In particular, the bags that contain an element from P make up a path in T'. The width is 3 since bags have size at most 4. Moreover, the decomposition is strict since the cones of the nodes cover ever larger subpaths due the construction. Every component $\alpha(t)$ for $t \in V(T')$ is the vertex set of a subpath of P and, thus, induces a substructure with a connected Gaifman graph; that means, D is normal.

We want $\varphi^\star$ to (1) define a tree decomposition of the shape described above, and, based on this, (2) mimic the behavior of φ on T. We start with the first requirement. A first part of the formula $\varphi^\star$ ensures that the tree underlying the decomposition is binary, leaves have singleton bags that partition the set of elements of P, and inner nodes have bags of size 4 and are constructed in the way described above—their cones are subpaths and they split

into two subpaths that are, in turn, the cones of the two child nodes. A second part of the formula accesses the tree T' underlying D and reconstructs the partition of children into left successors and right successors. This can be done by also using the successor relation of P. In total, this reconstructs all the information we need to simulate the behavior of φ on T.

Correctness follows from two observations: First, every tree T whose yield is P can be transformed into a tree decomposition D of the form described above and $T \models \varphi$ implies $(P, D) \models \varphi^\star$. Moreover, a tree decomposition D of the kind defined by the first part of $\varphi^\star$ can be turned back into a tree T that yields P, such that $(P, D) \models \varphi^\star$ implies $T \models \varphi$. ◀

▶ **Lemma 9.** *For every $w \in \mathbb{N}$ and* MSO*-sentence $\varphi^\star$ over $\tau^\star_{\Sigma\text{-paths}}$, there is an* MSO*-sentence φ over $\tau_{\Sigma\text{-trees}}$ satisfying the following for every Σ-path P: There is a width-w normal tree decomposition D for P with $(P, D) \models \varphi^\star$ if, and only if, there is a Σ-tree T that yields P with $T \models \varphi$.*

Proof. We start to give a transformation of normal tree decompositions for paths into trees that yield paths. Then we show how this transformation can be used to prove the lemma.

Let $D = (T, \beta)$ be a width-w normal tree decomposition for P. Due to the normalization, we know that every component $\alpha(t)$ for $t \in V(T)$ is connected and, since P is a path, it is a connected subpath P_t of P. Due to the strictness condition, the subpaths get larger when moving from a child to its parent or the bag size grows. Consider a node t of T and its children c_1 to c_ℓ, and let P_t and P_1 to P_ℓ be the paths that are induced by the respective components. Distinct paths P_i and P_j for $i, j \in \{1, \ldots, \ell\}$ have distinct elements and, using the at most $w + 1$ elements from $\beta(t)$, they are connected to form a path P_t. In particular, this means that ℓ is upper bounded in terms of w. To turn D into a tree T' that yields P we start with the tree T and insert additional nodes and edges to make it binary. First of all, we call the nodes of T *decomposition nodes*. Every decomposition node t is replaced by a chain of $|\beta(t) \setminus \sigma(t)| + \ell$ nodes; the chain is based on using binary relations from $\textsc{right}(T')$. The beginning of the chain is connected to the parent of t. Using binary relations from $\textsc{left}(T')$, we add edges to leaf nodes for the elements from $\beta(t) \setminus \sigma(t)$ and edges to the child nodes c_1 to c_ℓ to this chain. The order of adding edges is based on the order of the nodes and subpaths on the path P. We call the newly established leaves *element nodes*. The resulting binary ordered tree T' yields P via traversing the leaves based on the ordering that is induced by the tree.

For turning $\varphi^\star$ into φ, we first use subformulas that single out the decomposition and element nodes, respectively. This enables us to reconstruct the tree decomposition D from T'. Thus, we have access to all of (P, D) and define the property defined by $\varphi^\star$.

Correctness follows from the fact that φ over Σ-trees defines exactly the Σ-trees that encode decompositions as described above. Thus, we can move from Σ-trees to tree decompositions and back while maintaining the answer to the model relation for $\varphi^\star$ and φ. ◀

4.2 Beyond Context-Free Languages via General Decompositions

The logic DMSO is based on normal tree decompositions instead of general ones. The main reason behind this is based on the fact that, using general tree decompositions of bounded width, it is possible for an MSO-formula to define properties of strings that are not context-free. This is formally stated by the following lemma.

▶ **Lemma 10.** *Let $\Sigma = \{a, b, c\}$ and $L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$. There is an* MSO*-formula φ satisfying the following for every Σ-path P: we have $P \in \mathcal{P}(L)$ if, and only if, there is a width-3 tree decomposition D for P with $(P, D) \models \varphi$.*

Proof. The tree decompositions that are used to prove the lemma have the shape of a path. The root bag contains the left-most a-labeled node, the right-most b-labeled node and the left-most c-labeled node. Starting from these nodes, the a-labeled, b-labeled, and c-labeled subpaths are processed by visiting one node after the other in an alternating fashion. This means that the a-labeled nodes are processed from left to right, the b-labeled nodes are processed from right to left, and the c-labeled nodes are processed from left to right. In order to move from one node to the other, we add a single node to the current bag and, in the next step, this node replaces the prior one with the same label. The bags of the resulting decomposition have size at most 4. The properties of the decomposition, which are described above, are MSO-definable and the alternating replacement of nodes can be utilized to test whether the given path encodes a string $a^n b^n c^n$ for some $n \in \mathbb{N}$. ◀

The language L used in the previous lemma is not context-free with respect to the classical definition of context-free languages in terms of context-free grammars [13]. Nevertheless, other definitions of context-freeness in the context of graph grammars turn L into a context-free language [5].

5 Model-Theoretic Aspects

The logic DMSO on strings defines exactly the context-free languages as shown in the previous section. In the present section, we further the understanding of its expressive power for general structures. We start to show that its closure properties are similar to the ones of the context-free languages (Section 5.1). A DMSO-formula consists of a decomposition quantifier along with an MSO-formula that has access to both the structures and a tree decomposition for it. Since the expressive power of MSO-logic is well-understood (see, for example, [15] and [5]), we concentrate on the key aspect that DMSO-logic adds to MSO-logic: how the decomposition quantifier in conjunction with the built-in width bound influences the expressive power. We show that, already on graphs of tree width 1, increasing the width bound results in a higher expressive power (Section 5.2).

5.1 Closure of Definable Properties Under Set Operations

In the following, we develop properties that support DMSO-logic in being a reasonable logic-based generalization of the notion of context-freeness from strings to general structures. DMSO-logic has closure properties similar to the context-free languages and they exist for similar reasons. Moreover, it has an alternative characterization that is similar to the one of Chomsky and Schützenberger for the context-free languages.

▶ **Lemma 11.** *There is a vocabulary τ and* DMSO*-definable properties of τ-structure $\mathcal{P}$, $\mathcal{P}_1$, and $\mathcal{P}_2$, such that (complement) $\overline{\mathcal{P}}$ and (intersection) $\mathcal{P}_1 \cap \mathcal{P}_2$ are not* DMSO*-definable.*

Proof. To prove the lemma, we use $\tau_{\Sigma\text{-paths}}$ with $\Sigma = \{a, b, c\}$ and sets of Σ-paths $\mathcal{P} := \mathcal{P}(\{a^\ell b^m b^n \mid \ell \neq m \text{ or } m \neq n\})$, $\mathcal{P}_1 := \mathcal{P}(\{a^m b^n c^n \in \Sigma^* \mid m, n \in \mathbb{N}\})$, and $\mathcal{P}_2 := \mathcal{P}(\{a^m b^m c^n \in \Sigma^* \mid m, n \in \mathbb{N}\})$. The properties $\mathcal{P}$, $\mathcal{P}_1$, and $\mathcal{P}_2$ are DMSO-definable since they are based on context-free languages and we have Theorem 6. The complement of $\mathcal{P}$ restricted to Σ-paths is $\mathcal{P}' = \mathcal{P}(L)$ with language $L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$. Since L is not context-free (see, for example, Harrison's book [13] for a proof), Theorem 6 implies that $\overline{\mathcal{P}}$, the complement property of $\mathcal{P}$, is not DMSO-definable. Since $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathcal{P}(L)$, where L is the just defined language, which is not context-free, the non-closure with respect to intersection holds for the same reason. ◀

While standard closure properties of the context-free languages follow, for example, from the link to regular tree languages, which we used in Section 4, the closure properties of DMSO-logic follow from the syntax of their formulas.

▶ **Lemma 12.** *Let τ be a vocabulary, $\mathcal{P}_1$ and $\mathcal{P}_2$ DMSO-definable properties of τ-structures, and $\mathcal{P}$ an MSO-definable property of τ-structure. Then (union) $\mathcal{P}_1 \cup \mathcal{P}_2$ and (intersection with an MSO-property) $\mathcal{P}_1 \cap \mathcal{P}$ are DMSO-definable.*

Proof. Let $\psi_1 = \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi_1]$ and $\psi_2 = \exists D\,[\varphi_{\text{WIDTH-}w_2\text{-TD}} \wedge \varphi_2]$ be the DMSO-formulas that define $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, and φ be the MSO-formula that defines $\mathcal{P}$.

(union) Without loss of generality assume $w_1 \geq w_2$. To define the union $\mathcal{P}_1 \cup \mathcal{P}_2$, we use the formula $\psi' := \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi']$ with $\varphi' := \varphi_1 \vee [\varphi_{\text{WIDTH-}w_2\text{-TD}} \wedge \varphi_2]$. That means, the tree decomposition that we use for the inner part of the formula has width at most w_1 and satisfies φ_1 or it has a, possibly smaller, width w_2 and satisfies φ_2. This defines exactly the union of $\mathcal{P}_1$ and $\mathcal{P}_2$.

(intersection with an MSO-property) To define the intersection of $\mathcal{P}_1$ and $\mathcal{P}$, we use the formula $\psi' := \exists D\,[\varphi_{\text{WIDTH-}w_1\text{-TD}} \wedge \varphi']$ with $\varphi' := \varphi_1 \wedge \varphi^{\text{relativized}}$. In this definition, $\varphi^{\text{relativized}}$ arises from φ by relativizing each quantifier of the formula to only take elements from the given structure A into account, and not elements from the added decomposition D. Consequently, ψ' defines $\mathcal{P}_1 \cap \mathcal{P}$ since $\varphi^{\text{relativized}}$ only depends on A. ◀

DMSO-logic has an alternative definition similar to a theorem of Chomsky and Schützenberger (see, for example, Harrison's book [13] for background and a proof of it) about representing a context-free language as the strings generated by (1) taking words with just brackets that are well-formed, (2) take the words of this kind that satisfy a regular property, and (3) the image of these words under a homomorphism. In the below proposition, the words of brackets are replaced by structures expanded with tree decompositions, the particular type of a bracket in combination with the regular property is replace by an MSO-property, and the homomorphism is replaced by projecting from (A, D) to A. It can be seen as moving the interpretation of the existential decomposition quantifier into the structures defined by an MSO-formula via using a larger vocabulary.

▶ **Proposition 13.** *Let $\varphi^\star$ be an MSO-formula over $\tau^\star$ for some τ that defines a property $\mathcal{P}^\star$ of structures expanded by normal tree decompositions of a bounded width $w \in \mathbb{N}$. Then the property $\mathcal{P}$ that contains all τ-structures that arise from reducing structures in $\mathcal{P}^\star$ to the relations in τ is DMSO-definable.*

5.2 Influence of the Width Parameter on Definable Properties

We denote by SO the class of all properties that are SO-definable and by SO *on* $\mathcal{C}$ the class of properties that are SO-definable on a set $\mathcal{C}$ of structures. In the same way, we define classes of properties based on MSO-definability and DMSO-definability. For structures of bounded tree width, DMSO-logic relates to the other logics discussed as follows.

▶ **Theorem 14.** *Consider a width bound $w \geq 1$ and let $\mathcal{C}$ be the set of all graphs (structures over τ_{graphs}) with tree width at most w. Then* MSO $\subsetneq$ DMSO $\subsetneq$ SO *on* $\mathcal{C}$.

Proof. MSO is separated from DMSO on strings via Fact 5 and Theorem 6; that means, by proving that languages like $a^n b^n$ are context-free, but not regular. DMSO is separated from SO on strings via languages that are not context-free, but SO-definable like $a^n b^n c^n$. The latter follows from Lemma 10 and the observation that the used tree decomposition can be

represented by a relation, which can be guessed by a second-order existential quantifier. The theorem follows from the insight that a string over characters from a fixed alphabet can be represented by a path graph without unary relations for the characters, but with spikes of varying length leaving the path's nodes to encode the characters. ◀

In the following, we take a closer look at the expressive power of DMSO-formulas and how it varies for different kinds of formulas. Remember that a DMSO-formula has the form $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$. Since the expressive power of the subformula φ, which is evaluated for a structure of bounded tree width, can be understood from the large literature on MSO's expressive power (see [15] and [5] for an overview), we concentrate on understanding the influence of the width bound w on the expressive power of formulas. First of all, since the width of the tree decomposition can also be reduced using the subformula φ, a higher built-in width bound w never results in a weaker expressive power. On the other side, in the case of strings, the equivalence to the context-free languages holds for any width bound that is at least 3. That means, width bounds beyond 3 do not result in a higher expressivity on strings. In the following, we show that this behavior does not translate to more general structures. That means, in contrast to the situation on strings, some properties of structures with a constant tree width only become DMSO-definable by increasing the width bound of the used decompositions—we cannot bound their width in terms of the width of the input structures. Interestingly, subdivided star graphs are all we need to exhibit this behavior. They have only tree width 1 and path width 2, which means that they admit width-2 tree decompositions whose underlying tree is a path.

▶ **Theorem 15.** *There is a set of structures $\mathcal{C}$ with tree width 1 and for every $w \geq 3$ a* DMSO*-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$, such that there is no* DMSO*-formula $\psi' = \exists D\,[\varphi_{\text{tw-}w'\text{-dec}} \wedge \varphi']$ with $w' \leq w - 2$ where ψ and ψ' are equivalent on C.*

The proof of Theorem 15 uses types. The *(quantifier) rank* of an MSO-formula φ, denoted by $\operatorname{qr}(\varphi)$, is the maximum number of nested quantifiers in φ. The *rank-q type* of a structure A over some vocabulary τ and a tuple $\bar{a} \in U(A)^k$ for some $k \in \mathbb{N}$ is the set of MSO-formulas $\varphi(x_1, \ldots, x_k)$ over τ of rank at most q with $A \models \varphi(\bar{a})$; we denote it by $\operatorname{tp}_q(A, \bar{a})$ and also write $\operatorname{tp}_q(A)$ for $k = 0$. The class of rank-q types for a vocabulary τ and $k \in \mathbb{N}$ is

$$\operatorname{TP}_q(\tau, k) := \{\operatorname{tp}_q(A, \bar{a}) \mid A \text{ is a } \tau\text{-structure and } \bar{a} \in U(A)^k\}\,.$$

A type only contains a finite number (depending on τ, q, and k) of non-equivalent formulas and, thus, it can be represented by a finite set of formulas; one formula for each equivalence class of formulas (see [15] for more details). Consequently, we view types as constant-size objects and algorithmic operations on them run in constant time.

We apply the below composition theorem (Fact 16) showing how the MSO-type of a structure arises from the MSO-types of substructures that are combined by identifying elements (see the survey [12] for more details on this fact). The upcoming proof does not use the algorithmic part of the fact's statement, but we use it in Section 6.

▶ **Fact 16.** *Consider a vocabulary τ, arities $k, \ell, m \in \mathbb{N}$, and a rank bound $q \in \mathbb{N}$. There is an algorithm that outputs $\operatorname{tp}_q(A \cup B, \bar{w})$ on input of $\operatorname{tp}_q(A, \bar{u}\bar{w})$ and $\operatorname{tp}_q(B, \bar{v}\bar{w})$ where A and B are τ-structures with tuples $\bar{u} = (u_1, \ldots, u_k) \in A^k$, $\bar{v} = (v_1, \ldots, v_\ell) \in B^\ell$, and $\bar{w} = (w_1, \ldots, w_m) \in (A \cap B)^m$ with $A \cap B = \{w_1, \ldots, w_m\}$.*

Proof of Theorem 15. The class $\mathcal{C}$ contains all subdivisions of stars. A star is a tree where all leaves are adjacent to the root and a subdivision of it arises by replacing edges with paths;

we call them *stars* for sake of simplicity. They have tree width 1 and, moreover, also path width 2. The paths that leave the root are called *rays*. Out of them, it will be enough to look at stars where the number of rays is bounded by the width parameter w from the theorem's statement. Formally, we use the following properties of graphs: STARS := {graph G | G is a subdivided star graph}, w-STARS := {graph G | G is a subdivided star with w rays}. The separating query singles out stars with rays of equal length: STARS-EQU := {graph G | $G \in$ STARS and all rays have the same length} and w-STARS-EQU := w-STARS $\cap$ STARS-EQU

For $w \in \mathbb{N}$ with $w \geq 3$, we claim that there is a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ defining w-STARS-EQU, but no DMSO-formula $\psi' = \exists D\,[\varphi_{\text{tw-}w'\text{-dec}} \wedge \varphi']$ with $w' \leq w-2$ defining w-STARS-EQU. This claim proves the theorem.

For the first part of the claim, φ consists of several parts combined by a conjunction; $\varphi := \varphi_{\text{STARS}} \wedge \varphi_{w\text{-LEAVES}} \wedge \varphi_{\text{EQU}}$. The MSO-formula φ_{STARS} defines the set of all subdivided stars while $\varphi_{w\text{-LEAVES}}$ defines the set of graphs with at most w degree-1 vertices. If these properties are satisfied, we are dealing with a subdivided star that has at most w rays. Based on this, the formula ψ_{EQU} accesses a decomposition D to single out the subdivided stars that have rays of equal length. First of all, it restricts D to be a tree decomposition whose underlying tree is a path (called a *path decompositions*), such that the left-most bag contains all leaves of the subdivided star. Then edges of the rays are added in an alternating way until also all edges to the root are covered. Consuming the edges in an alternating way ensures that the rays have equal length. MSO-definability of these requirements holds since w is constant.

Let $\psi' = \exists D\,[\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi']$ be a DMSO-formula that, for sake of a contradiction, defines w-STARS-EQU. Consider a subdivided star graph G with w rays, let D be a normal tree decomposition of width at most $w-2$, and let v be the root of the star. Let B_v be the highest bag in D that contains v and let v_1 to $v_{w'}$ for $w' \leq w-2$ be the other vertices of G in B_v. The bag B_v is not a leaf since, otherwise, there are edges leaving v that are not covered by the tree decompositions. Due to the same reason, there are child bags (that means, at least one child bag) that contain v. At least two rays of the star are processed by decompositions that are rooted at these children and that are disjoint except for, possibly, the root bag. This follows from the normalization condition, which makes all components defined by the tree decomposition connected. Let B_1 and B_2 be child bags of B_v (that may be the same) and D_1 and D_2 be tree decompositions inside D of rays P_1 and P_2 of the star with roots B_1 and B_2. We assume that D_2 together with P_2 is large enough, such that the smallest structure of the same rank-qr($\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi'$) type over $\tau^{\star}_{\text{graphs}}$ covers a graph (and, hence, a path) with fewer nodes. We replace (P_2, D_2) in (G, D) by this structure. The resulting structure (G', D') still satisfies $\varphi_{\text{tw-}(w-2)\text{-dec}} \wedge \varphi'$ by the construction and the composition theorem from Fact 16 with choosing $k, \ell, m \in \mathbb{N}$ appropriately, but not all rays of G' have the same length. Thus, ψ' does not define w-STARS-EQU, a contradiction. ◀

6 Algorithmic Aspects

The satisfiability problem [17] and evaluation problem [4, 7] for MSO-logic on tree-width-bounded structures are well-studied. We extend this work by studying the decidability of the satisfiability and related problems for DMSO-logic (Section 6.1) and the complexity of its formula-evaluation problem (Section 6.2). In order to do this, formulas and structures need to be represented as strings—to be given as inputs, processed by Turing machines, and written as outputs. A formula is encoded in a direct way by encoding its symbols individually and a structure is encoded by extending the standard adjacency matrix encoding for graphs to general structures (for details of this approach see, for example, [10]).

6.1 Decidability of the Satisfiability and Related Problems

Trakhtenbrot's Theorem (for a proof see [15]) states that it is undecidable whether a given first-order formula is satisfied by a finite structure. Thus, this also transfers to MSO-logic, which generalizes first-order logic. On the other side, part of Seese's Theorem [17] is saying that the satisfiability problem for MSO-formulas is decidable if there is a bound on the tree width of the considered structures. This can be formally stated as follows.

▶ **Fact 17.** *There is a Turing machine that, given an* MSO*-formula φ over a vocabulary τ and a width $w \in \mathbb{N}$, decides whether there is a τ-structure A with* $\mathrm{tw}(A) \leq w$ *and* $A \models \varphi$.

Since DMSO-formulas are based on combining a tree width bound with an MSO-formula, Seese's Theorem extends to them. In light of the connection to context-free languages, this can be seen as generalizing the fact that the emptiness problem for context-free languages (given a context-free grammar or a pushdown automaton) is decidable.

▶ **Theorem 18.** *There is a Turing machine that, given a* DMSO*-formula ψ over some vocabulary τ, decides whether there is a τ-structure A with $A \models \psi$.*

Proof. Let $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$. As stated by Lemma 2, a structure A that is expanded by a tree decomposition of width at most w has tree width at most $w + 2$. Thus, in order to decide whether ψ is satisfied by a τ-structure, it is enough to decide whether the inner part of the formula, which is $\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi$, is satisfied by a $\tau^\star$-structure of tree width at most $w + 2$. This is possible due to Fact 17. ◀

Since MSO-logic is closed under taking Boolean connectives, the decision procedure that we get from Seese's Theorem extends to the question whether any Boolean combination of two given MSO-formulas is satisfied by a structure of bounded tree width. Due to its equivalence on strings to the context-free languages, these closure properties do not transfer to DMSO-logic. DMSO-logic inherits the following undecidability results from the context-free languages (for proofs of them see, for example, [13]).

▶ **Proposition 19.** *Each of the following questions is not decidable for given* DMSO*-formulas ψ_1 and ψ_2 defining properties $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively: $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$? $|\mathcal{P}_1 \cap \mathcal{P}_2| = \infty$? $\mathcal{P}_1 \cap \mathcal{P}_2 \in$ DMSO? $\mathcal{P}_1 \subseteq \mathcal{P}_2$? $\mathcal{P}_1 = \mathcal{P}_2$?*

6.2 Computational complexity of the Evaluation Problems

The task of evaluating an MSO-formula for a structure A comes in many flavors; leading to different insights into the complexity of evaluating MSO-formulas. If both the formula and the structure are part of the input, then the problem is complete for polynomial space [20]. If the formula is fixed (that means, we look at a problem based on an MSO-definable property), then we still have problems complete for any level of the polynomial hierarchy (see [15] for a proof of this). The complexity of the problem drops significantly if, in addition, the tree width of input structures is bounded by some constant: deciding $A \models \varphi$ can be done in polynomial time for input structures whose tree width is bounded by some constant and even algorithms running in linear-time [4] or having a logarithmic memory footprint exist [7]. During the course of the present section, we show that deciding $A \models \psi$ can also be done in polynomial time for every fixed DMSO-formula ψ (where we do not need to make the tree width bound explicit since it is implicitly given as part of the formula).

▶ **Theorem 20.** *For every* DMSO*-formula ψ, there is a polynomial-time algorithm that, given a structure A, decides $A \models \psi$.*

Proof. We consider a DMSO-formula $\psi = \exists D\,[\varphi_{\text{tw-}w\text{-dec}} \wedge \varphi]$ over some vocabulary τ and set $q := \mathrm{qr}(\varphi)$. While $\exists D$, the decomposition quantifier, ranges over all tree decompositions of a given structure A, the MSO-formula $\varphi_{\text{tw-}w\text{-dec}}$ only evaluates to true for structures expanded by a tree decompositions that is normal and has width w. Thus, in order to test whether $A \models \psi$ holds for a given τ-structure, it is enough to test whether $(A, D) \models \varphi$ holds for some width-w normal tree decomposition D of A. Instead of directly working with MSO-formulas, we work with types as already used in Section 5. That means, instead of testing whether $(A, D) \models \varphi$ holds for a width-w normal tree decompositions, we compute

$$\mathrm{TP}^\star_q(A) := \{\mathrm{tp}_q(A^\star) \mid A^\star = (A, D) \text{ is a } \tau^\star\text{-structure with } (A, D) \models \varphi_{\text{tw-}w\text{-dec}}\},$$

which has constant size by definition. Then deciding $A \models \psi$ is equivalent to testing whether φ is equivalent to a formula from θ for some $\theta \in \mathrm{TP}^\star_q(A)$. Since this is a constant-time operation, we direct our attention to computing $\mathrm{TP}^\star_q(A)$.

We compute $\mathrm{TP}^\star_q(A)$ via an inductive approach along growing substructures. It computes types for substructures of A by applying the type-composition theorem from Fact 16 to the types that are already computed for smaller substructures. Since the types are based on both the structure and a tree decomposition for it, while doing this, we enrich the substructures of A by parts of candidate tree decompositions that expand them. Overall, we take the MSO-types of all width-w normal tree decompositions for A into account, but without constructing decompositions explicitly; this amounts to a dynamic-programming approach along appropriate substructures.

We use some terminology adapted from [7] to define the substructures and the types of substructures we consider: Set $G := G(A)$ and $U := U(A)$. A *descriptor* in A is a tuple (σ_i, v_i) consisting of a set $\sigma_i \subseteq U$ with $|\sigma_i| \leq w + 1$, called the *separator*, and an element $v_i \in U \setminus \sigma_i$, called the *(component) selector*. We denote by α_i the vertex set of the component of $G[U \setminus \sigma_i]$ that contains v_i and set $\gamma_i := \sigma_i \cup \alpha_i$. For each (σ_i, v_i) with $A_i := A[\gamma_i]$, we consider some (ordered) sequence $\bar{\sigma}_i$ of the elements in σ_i and compute

$$\begin{aligned}\mathrm{TP}^\star_q(A_i, \sigma_i) := \{\mathrm{tp}_q(A^\star_i, \bar{\sigma}^\star_i) \mid A^\star_i = (A_i, D_i) \text{ is a } \tau^\star\text{-structure with} \\ (A_i, D_i) \models \varphi_{\text{tw-}w\text{-dec}}, \sigma_i \subseteq \beta(\mathrm{r}(D_i)) \text{ and } \bar{\sigma}^\star_i = (\bar{\sigma}_i, \mathrm{r}(D_i))\},\end{aligned}$$

which is only empty if there is no width-w normal tree decomposition of the described kind. Since $G[A]$ is connected (otherwise, A does not have a normal tree decomposition by definition), we have $\mathrm{TP}^\star_q(A) = \mathrm{TP}^\star_q(A_{i_0}, \bar{\sigma}_{i_0})$ for a descriptor (σ_{i_0}, v_{i_0}) with $\sigma_{i_0} = \emptyset$ and an arbitrary, but fixed, element $v_{i_0} \in U$. Thus, in particular, computing all $\mathrm{TP}^\star_q(A_i, \bar{\sigma}_i)$ for descriptors (σ_i, v_i) proves the theorem.

Let $(\sigma_1, v_1), \ldots, (\sigma_n, v_n)$ be the sequence of all descriptors sorted by $|\gamma_i|$, the size of their cones, with higher priority and by $|\alpha_i|$, the size of their components, with lower priority. This sequence is computable in polynomial time since w is constant. We compute $\mathrm{TP}^\star_q(A_i, \bar{\sigma}_i)$ for each (σ_i, v_i) in the order of this sequence. If $|\gamma_i| \leq w + 1$, we can compute it in constant time; there are only a constant number of ways to turn the structure A_i into a structure that is expanded by a width-w normal tree decomposition. If $|\gamma_i| > w + 1$, we consider each $\beta_{i,j} \subseteq \gamma_i$ with $\sigma_i \subseteq \beta_{i,j}$ and $|\beta_{i,j}| \leq w + 1$, which are the candidate root bags of a tree decomposition for A_i, and compute

$$\begin{aligned}\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j}) := \{\mathrm{tp}_q(A^\star_i, \bar{\sigma}^\star_i) \mid A^\star_i = (A_i, D_i) \text{ is a } \tau^\star\text{-structure with} \\ (A_i, D_i) \models \varphi_{\text{tw-}w\text{-dec}}, \beta_{i,j} = \beta(\mathrm{r}(D_i)) \text{ and } \bar{\sigma}^\star_i = (\bar{\sigma}_i, \mathrm{r}(D_i))\}.\end{aligned}$$

Then $\mathrm{TP}^\star_q(A_i, \sigma_i)$ can be computed in polynomial time from all $\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j})$ by cycling through the candidate root bags $\beta_{i,j}$ of size at most $w + 1$.

For each $\beta_{i,j}$, we consider the components $\alpha'_1, \ldots, \alpha'_m$ of $A_i \setminus \beta_{i,j}$ and define $A'_m = A_i$ as well as $A'_k = A'_{k+1} \setminus \alpha'_{k+1}$ for all $k \in \{1, \ldots, m-1\}$. With this definition, we have $\mathrm{TP}^\star_q(A_i, \sigma_i, \beta_{i,j}) = \mathrm{TP}^\star_q(A'_m, \sigma_i, \beta_{i,j})$. In order to compute this class, we compute the $\mathrm{TP}^\star_q(A'_k, \sigma_i, \beta_{i,j})$ along growing k. For each k with component α'_k, we consider all combinations of separators $\sigma'_{i,j,k} \subseteq \beta_{i,j}$ and selector vertices $v'_{i,j,k} \in \alpha'_j$ that describe this component. Then we use the type-composition theorem to compute the types of A'_k that we get from normal width-w tree decompositions where the intersection of the root bag $\beta_{i,j}$ and the bag below it with component α'_k is $\sigma'_{i,j,k}$. Fact 16 can be applied since the overlap of A'_{k-1} and the newly added part has constant size and this still holds if we take the two additional nodes from the tree decomposition into account. Since the type-composition is a constant-time operation, the algorithm runs in polynomial time. ◀

7 Conclusion

The present paper proposed DMSO-logic for capturing the notion of context-free graph properties. To support this, we proved results about its language-theoretic, model-theoretic, and algorithmic aspects. Notably, it corresponds to the context-free languages on strings and has a formula-evaluation procedure that runs in polynomial time for every fixed formula. Moreover, we observed that the built-in width bound crucially influences expressive power.

There are a number of possible directions for future work: First of all, it would be interesting to know how DMSO-logic relates to the different notions of context-free properties of trees [9] and, more generally, graphs [5]. Second, it would be interesting to obtain more efficient formula-evaluation procedures. Concrete open questions are whether deciding $A \models \psi$ is fixed-parameter tractable when taking ψ as the parameter and whether it is in LOGCFL (the class of problems logspace-reducible to the context-free languages) for every fixed formula. Both are reasonable conjectures in the light of previous work on the formula-evaluation problem for MSO-logic [4, 7].

References

1. Isolde Adler, Martin Grohe, and Stephan Kreutzer. Computing excluded minors. In *Proceedings of the 19th Annual ACM/SIAM Symposium on Discrete Algorithms (SODA 2008)*, pages 641–650. SIAM, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347153`.
2. Mikołaj Bojańczyk and Michał Pilipczuk. Definability equals recognizability for graphs of bounded tree width. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2016)*. IEEE Computer Society, 2016. to appear.
3. J. Richard Büchi. Weak second-order arithmetic and finite automata. *Math. Logic Quart.*, 6(1–6):66–92, 1960.
4. Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Handbook of Theoretical Computer Science, Volume B*, pages 193–242. Elsevier and MIT Press, 1990.
5. Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic, a language theoretic approach*. Cambridge University Press, 2012. URL: `http://hal.archives-ouvertes.fr/hal-00646514/fr/`.
6. John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970. `doi:10.1016/S0022-0000(70)80041-1`.
7. Michael Elberfeld, Andreas Jakoby, and Till Tantau. Logspace versions of the theorems of bodlaender and courcelle. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*, pages 143–152, 2010. `doi:10.1109/FOCS.2010.21`.

8 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961. doi:10.2307/1993511.

9 Joost Engelfriet and Erik Meineche Schmidt. IO and OI. I. *Journal of Computer and System Science*, 15(3):328–353, 1977. doi:10.1016/S0022-0000(77)80034-2.

10 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

11 Martin Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *Journal of the ACM*, 59(5):27:1–27:64, 2012. doi:10.1145/2371656.2371662.

12 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. In *Model Theoretic Methods in Finite Combinatorics*, volume 558 of *Contemporary Mathematics*, pages 181–206. American Mathematical Society, 2011. URL: http://www.automata.rwth-aachen.de/~grohe/pub/grokre11.pdf.

13 Michael A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

14 Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In *Proceedings of CSL 1994*, volume 933, pages 205–216. Springer, 1995.

15 Leonid Libkin. *Elements Of Finite Model Theory*. Springer, 2004. doi:10.1002/malq.19600060105.

16 J. Mezei and J.B. Wright. Algebraic automata and context-free sets. *Inform. and Control*, 11(1–2):3–29, 1967. doi:10.1016/S0019-9958(67)90353-1.

17 D. Seese. The structure of the models of decidable monadic theories of graphs. *Annals of pure and applied logic*, 53(2):169–195, 1991. doi:10.1016/0168-0072(91)90054-P.

18 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. doi:10.1007/BF01691346.

19 Boris Avraamovich Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 140:326–329, 1961. In Russian.

20 Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the 14th annual ACM symposium on Theory of computing (STOC 1982)*, pages 137–146. ACM, 1982. doi:10.1145/800070.802186.

Successor-Invariant First-Order Logic on Graphs with Excluded Topological Subgraphs

Kord Eickmeyer[1] **and Ken-ichi Kawarabayashi**[2]

1 **Technical University Darmstadt, Department of Mathematics, Schlossgartenstr. 7, 64289 Darmstadt, Germany**
eickmeyer@mathematik.tu-darmstadt.de

2 **National Inst. of Informatics, Tokyo, Japan; and JST, ERATO, Kawarabayashi Large Graph Project, Chiyoda-ku, Hitotsubashi 2-1-2, Tokyo 101-8430, Japan**
k_keniti@nii.ac.jp

Abstract

We show that the model-checking problem for successor-invariant first-order logic is fixed-parameter tractable on graphs with excluded topological subgraphs when parameterised by both the size of the input formula and the size of the exluded topological subgraph. Furthermore, we show that model-checking for order-invariant first-order logic is tractable on coloured posets of bounded width, parameterised by both the size of the input formula and the width of the poset.

Results of this form, i.e. showing that model-checking for a certain logic is tractable on a certain class of structures, are often referred to as algorithmic meta-theorems since they give a unified proof for the tractability of a whole range of problems. First-order logic is arguably one of the most important logics in this context since it is powerful enough to express many computational problems (e.g. the existence of cliques, dominating sets etc.) and yet its model-checking problem is tractable on rich classes of graphs. In fact, Grohe et al. [21] have shown that model-checking for FO is tractable on all nowhere dense classes of graphs.

Successor-invariant FO is a semantic extension of FO by allowing the use of an additional binary relation which is interpreted as a directed Hamiltonian cycle, restricted to formulae whose truth value does not depend on the specific choice of a Hamiltonian cycle. While this is very natural in the context of model-checking (after all, storing a structure in computer memory usually brings with it a linear order on the structure), the question of how the computational complexity of the model-checking problem for this richer logic compares to that of plain FO is still open.

Our result for successor-invariant FO extends previous results for this logic on planar graphs [14] and graphs with excluded minors [13], further narrowing the gap between what is known for FO and what is known for successor-invariant FO. The proof uses Grohe and Marx's structure theorem for graphs with excluded topological subgraphs [22]. For order-invariant FO we show that Gajarský et al.'s recent result [19] for FO carries over to order-invariant FO.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases model-checking, algorithmic meta-theorem, successor-invariant first-order logic, topological subgraphs, parameterised complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.18

1 Introduction

Model-checking is one of the core algorithmic problems in finite model theory: Given a sentence φ in some logic L and a finite structure A, decide whether $A \models \varphi$. The problem

© Kord Eickmeyer and Ken-ichi Kawarabayashi;
licensed under Creative Commons License CC-BY

can be generalised by allowing φ to have free variables, in which case we would like to find instances $\bar{a}$ for which $A \models \varphi[\bar{a}]$, or count the number of such instances. One important application of this is the case where φ is a database query and A the database to be queried. The logic L from which φ is drawn then serves as an abstract model of the database query language.

Commonly studied logics L include first-order logic (FO) and monadic second-order logic (MSO). Even for first-order logic the model-checking problem is PSPACE complete already when restricted to structures A with two elements. On the other hand, for every fixed FO-formula φ, checking whether $A \models \varphi$ can be done in time polynomial in the size of A. This discrepancy between the *query complexity*, i.e. the complexity depending on the size of the query φ on the one hand and the *data complexity*, i.e. the complexity depending on the size of the structure A, on the other hand suggests that the complexity of model-checking problems is best studied in the framework of *parameterised complexity* [8, 17].

In parameterised complexity, apart from the size n of the input problem (commonly the length of an appropriate binary representation of φ and A) a *parameter* k is introduced. For model-checking problems the size of the input formula is a common choice of parameter. The role of PTIME as the class of problems commonly considered to be tractable is played by the parameterised complexity class of *fixed-parameter tractable (fpt)* problems, i.e. problems which can be solved in time

$$f(k) \cdot n^c$$

for some computable function f and a constant c. Note that the constant c must not depend on k, and indeed the model-checking problem for first-order logic is unlikely to be fixed-parameter tractable.

In order to obtain tractable instances of model-checking problems, one can restrict the space of admissible input structures A, e.g. by requiring the Gaifman graph of A to possess certain graph theoretic properties such as bounded degree or planarity. A long list of results have been obtained, starting with Courcelle's famous result that model-checking for monadic second-order logic is fixed-parameter tractable on structures A with bounded tree-width [4].

Results of this form are often referred to as *algorithmic meta-theorems* because many classical problems can be rephrased as model-checking problems by formalising them as a sentence φ in a suitable logic. For example, since the existence of a Hamiltonian cycle in a graph G of bounded tree-width can be expressed by a sentence φ of monadic second-order logic, Courcelle's Theorem immediately implies that hamiltonicity can be checked in polynomial time on such graphs. Besides giving a mere proof of tractability, algorithmic meta-theorems provide a unified treatment of how structural properties can be used in algorithm design. Cf. [20] and [24] for excellent surveys of the field of algorithmic meta-theorems.

The model-checking problem for first-order logic is particularly well studied and has been shown to be fixed-parameter tractable on a large number of graph classes: Starting with Seese's result [28] for graphs of bounded degree, Frick and Grohe showed tractability on classes of graphs with bounded tree-width and, more generally, locally bounded tree-width [18], which in particular includes planar graphs. This has been generalised to graph classes with excluded minors [16] and locally excluded minors [6]. Using rather different techniques, Dvořák et al. gave a linear fpt model-checking algorithm for first-order logic on graphs of bounded expansion [9]. As a generalisation of all the graph classes mentioned so far, Grohe et al. have shown in [21] that model-checking for first-order logic is possible in near-linear fpt on all nowhere dense graph classes.

While the tractability of model-checking for first-order logic on sparse graphs is well understood, few results are available for classes of dense graphs. Recently, Gajarský et al.

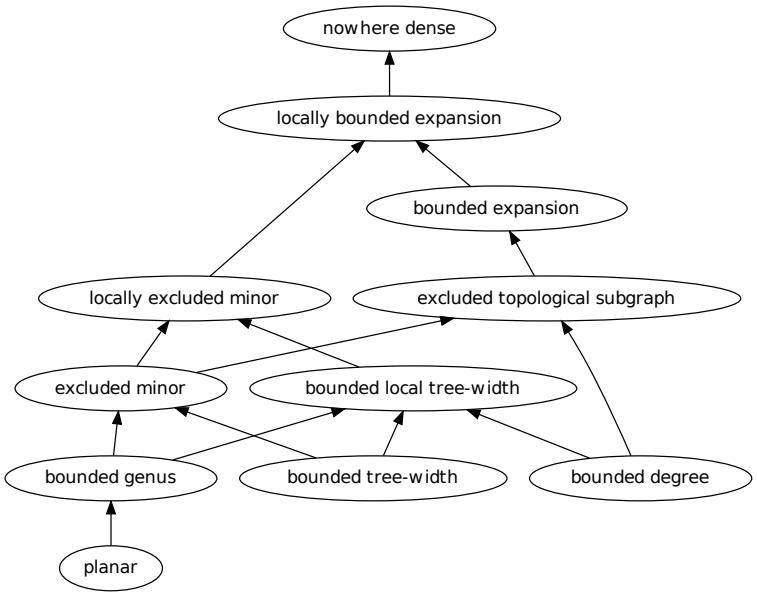


Figure 1 Sparse classes of graphs on which model-checking for first-order logic is tractable.

gave an fpt algorithm for FO model-checking on posets of bounded width, which we extend to order-invariant FO in Section 5.

Excluded Topological Subgraphs

A more general concept than that of a class of graphs excluding some graph H as a minor is that of graphs which exclude H as a *topological subgraph*. This is the concept originally used by Kuratowski in his famous result that a graph is planar if, and only if, it does not contain K_5 nor $K_{3,3}$ as a topological subgraph (cf. Section 4.4 in [7]). Recently, Grohe and Marx have extended Robertson and Seymour's graph structure theorem to classes of graphs excluding a fixed graph H as a topological subgraph [22]: These graphs can be decomposed along small separators into parts which exclude H as a minor and parts in which all but a bounded number of vertices have small degree.

Since every topological subgraph of a graph G is also a minor of G, if a class $\mathcal{C}$ of graphs excludes some graph H as a topological subgraph then it also excludes H as a minor. The converse is not true, however, since every 3-regular graph excludes K_5 as a topological subgraph, but for every $r \in \mathbb{N}$ there is a 3-regular graph containing K_r as a minor. On the other hand, graph classes with excluded topological subgraphs have bounded expansion, so model-checking for first-order logic is tractable on these classes by Dvořák et al.'s result.

Figure 1 shows an overview of sparse graph classes on which model-checking for first-order logic is tractable. Note that a class $\mathcal{C}$ of graphs excludes some finite graph H as a topological subgraph if, and only if, there is an $r \in \mathbb{N}$ such that $\mathcal{C}$ excludes the clique K_r as a topological subgraph.

Successor-Invariant Logic

We investigate the question in how far tractability results for first-order model-checking carry over to *successor-invariant first-order logic*, i.e. first-order logic enriched by a binary successor relation, restricted to formulae whose truth value does not depend on the specific choice of successor relation. Linear representations of an input structure A to a model-checking algorithm usually induce some linear order on the elements of $V(A)$, and it seems natural to make this linear order (or at least its successor relation) accessible to the query formula. This may, however, break the structural properties of the Gaifman graph of A needed by the model-checking algorithm.

Having access, even invariantly, to a successor relation provably increases the expressive power of FO on finite structures, as shown in [27]. However, all known classes of structures separating FO from order-invariant or successor-invariant FO contain large cliques, and in fact on trees [2] and on structures of bounded tree-depth [12] even order-invariant FO has the same expressive power as plain FO. On all the classes depicted in Figure 1, this question is still open, prompting for tractability results for successor-invariant or even order-invariant FO on these classes.

Previous work investigating the complexity of model-checking for successor-invariant first-order logic to that of plain first-order logic has been carried out by [14], who showed tractibility on planar graphs, and [13], who showed tractability on graph classes with excluded minors. Here we extend these results further by generalising from excluded minors to excluded topological subgraphs, further narrowing the gap between what is known for first-order logic and succesor-invariant first-order logic.

Note that for first-order logic, the result of [21] is optimal if one restricts attention to classes of graphs which are closed under taking subgraphs. In fact, Kreutzer has shown in [24] that under the complexity theoretic assumption that FPT $\neq$ W[1], if model-checking for FO on some subgraph-closed class $\mathcal{C}$ of graphs is fixed-parameter tractable, then $\mathcal{C}$ is nowhere dense (see also Section 1.4 of [9]). Examples of classes of graphs on which model-checking is fpt even for monadic second-order logic but which are not nowhere dense are graphs of bounded clique-width [5].

2 Preliminaries and Notation

For a natural number n we let $[n]$ denote the interval $\{1, \dots, n\}$.

2.1 Graphs

We will be dealing with finite simple (i.e. loop-free and without multiple edges) undirected graphs, cf. [7, 29] for an in-depth introduction. Thus a *graph* $G = (V, E)$ consists of some finite set V of *vertices* and a set $E \subseteq \binom{V}{2}$ of *edges*. We write $uv \in E$ for $\{u, v\} \in E$. For a set $U \subseteq V$ we denote the *induced subgraph* on U by $G[U]$, i.e. the graph (U, E') with

$$E' := \{uv \mid u, v \in U \text{ and } uv \in E\}.$$

For ease of notation we occasionally blur the distinction between a set U of vertices and the subgraph induced on this set. The union $G \cup H$ of two graphs $G = (V, E)$ and $H = (U, F)$ is defined as the graph $(U \cup V, E \cup F)$. For a set U of vertices, $K[U]$ denotes the complete graph (or *clique*) with vertex set U. For $k \in \mathbb{N}$, we denote the k-clique $K[[k]]$ by K_k.

A *walk* is a sequence of vertices $v_1, \dots, v_\ell \in V$, alternatively written as a function $v : [\ell] \to V$, such that $v_i v_{i+1} \in E$ for all $i = 1, \dots, \ell - 1$. A *path* is a walk in which

$v_i \neq v_j$ for $i \neq j$, except possibly $v_1 = v_\ell$, in which case the path is called a *cycle*. The vertices $v_2, \dots, v_{\ell-1}$ are called *inner vertices*. Two paths $v_1, \dots, v_\ell$ and $w_1, \dots, w_m$ are called *independent* if neither of them contains an inner vertex of the other, i.e. if $v_i = w_j$ implies $i \in \{1, \ell\}$ and $j \in \{1, m\}$.

For $k \geq 1$, a *k-walk* through a graph $G = (V, E)$ is a surjective walk $w : [\ell] \to V$ such that

$$1 \leq |\{i \in [\ell] \mid w(i) = v\}| \leq k$$

for all $v \in V$. A 1-walk is also called a *Hamiltonian path*.

Tree-Decompositions

A *tree* is a connected acyclic graph. A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, \mathcal{V})$ consisting of a tree $\mathcal{T} = (T, F)$ and a mapping $\mathcal{V} : T \to 2^V, t \mapsto \mathcal{V}_t$ such that

- $\bigcup_{t \in T} \mathcal{V}_t = V$,
- for every edge $uv \in E$ there is a $t \in T$ with $u, v \in \mathcal{V}_t$, and
- for every $v \in V$ the set $\{t \in T \mid v \in \mathcal{V}_t\}$ is a subtree of $\mathcal{T}$ (i.e. it is connected).

The sets $\mathcal{V}_t$ are called the *bags* of the tree-decomposition. Let $t \in \mathcal{T}$ have neighbours $\mathcal{N}(t) \subseteq \mathcal{T}$. The *torso* $\bar{\mathcal{V}}_t$ of $\mathcal{V}_t$ is the graph

$$G[\mathcal{V}_t] \cup \bigcup_{u \in \mathcal{N}(t)} K[\mathcal{V}_t \cap \mathcal{V}_u].$$

The graphs we will be dealing with do not in general allow tree-decompositions into bags of small size, but they do have decompositions $(\mathcal{T}, \mathcal{V})$ for which (the torsos of) all bags $\mathcal{V}_t$ have nice structural properties and for which

$$|\mathcal{V}_s \cap \mathcal{V}_t|$$

is small for all $s \neq t \in \mathcal{T}$. The *(maximal) adhesion* of $(\mathcal{T}, \mathcal{V})$ is the maximum of $|\mathcal{V}_s \cap \mathcal{V}_t|$ for all $s \neq t \in \mathcal{T}$.

Subgraphs, Minors, Topological Subgraphs

Let $G = (V, E)$ and $H = (W, F)$ be graphs. If $W \subseteq V$ and $F \subseteq E$ then we call H a *subgraph* of G and write $H \leq G$. In other words, H can be obtained from G be removing vertices and edges.

We say that H is a *minor* of G, written $H \preceq G$, if there are disjoint connected nonempty subgraphs $(B_w)_{w \in W}$ in G such that for every edge $xy \in F$ there is an edge $ab \in E$ for some $a \in B_x$ and $b \in B_y$. The sets $(B_w)_{w \in W}$ are called *branch sets* of the minor H. Equivalently, $H \preceq G$ if H can be obtained by repeatedly contracting edges in a subgraph of G.

A graph H' is a *subdivision* of a graph H if it can be obtained from H be replacing edges with paths. If $H' \leq G$ for some subdivision H' of H we say that H is a *topological subgraph* of G and write $H \preceq_{\text{top}} G$. In this case there is an injective mapping $\iota : W \to V$ and independent paths $P_{\iota(u)\iota(v)}$ connecting $\iota(u)$ to $\iota(v)$ in G for $uv \in F$. The vertices in the image of ι are called *branch vertices*. Obviously $H \preceq_{\text{top}} G$ implies $H \preceq G$, but the converse is not in general true.

2.2 Logics

We will be dealing with finite structures over finite, relational vocabularies. Thus a *vocabulary* σ is a finite set of relation symbols R, each with an associated *arity* $a(R)$, and a *σ-structure*

A consists of a finite set $V(A)$ (the *universe*) and relations $R(A) \subseteq A^{a(R)}$ for all $R \in \sigma$. For vocabularies $\sigma \subseteq \tau$ and a σ-structure A, a *τ-expansion* B is a τ-structure with $V(A) = V(B)$ and $R(B) = R(A)$ for all $R \in \sigma$.

The *Gaifman graph* of a structure A is the graph with vertex set $V(A)$ and edge set

$$\{xy \mid x \text{ and } y \text{ appear together in some relation } R(A)\}.$$

When applying graph-theoretic notions such as planarity to relational structures, we mean that the corresponding Gaifman graph has the said property.

We use standard definitions for first-order logic (FO), cf. [11, 10, 25]. In particular, $\bot$ and $\top$ denote false and true, respectively. Let σ be a vocabulary and $\text{succ} \notin \sigma$ a new binary relation symbol. We set $\sigma_{\text{succ}} := \sigma \cup \{\text{succ}\}$ and say that succ is interpreted by a *successor relation* in a σ_{succ}-structure B if $\text{succ}(B)$ is the graph of a cyclic permutation on $V(B)$. An $\text{FO}[\sigma_{\text{succ}}]$-formula φ is called *successor-invariant* if for all σ-structures A and all σ_{succ}-expansions B, B' of A in which succ is interpreted by a successor relation we have

$$B \models \varphi \quad \Leftrightarrow \quad B' \models \varphi,$$

when all free variables of φ are interpreted identically in B and B'. In this case we say that $A \models \varphi$ if $B \models \varphi$ for one such expansion B (equivalently for all such expansions).

Note that another common definition of successor relation is to require $\text{succ}(A)$ to be of the form

$$\{(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)\}$$

for some enumeration $V(A) := \{a_1, \ldots, a_n\}$ of the elements of $V(A)$. This differs from our definition in that we require $(a_n, a_1) \in \text{succ}(A)$ as well, eliminating the somewhat artificial status of the first and last element. This does not affect the expressive power of successor-invariant FO, because a cyclic successor relation can be obtained from a linear one using a simple FO interpretation and vice versa. Note that the quantifier rank of formulas is slightly increased by this interpretation.

Order-invariant first-order logic is defined analogously to successor-invariant FO, by allowing the use of a binary relation $\leq$ which is interpreted as a linear order and demanding the truth value of a formula to be independent of the chosen linear order.

3 Model-Checking for Successor-Invariant First-Order Logic

The main result of this paper is the following:

▶ **Theorem 1.** *There is an algorithm $\mathbb{A}$ which takes as input*

- *a finite graph H,*
- *a finite σ-structure A over some relational vocabulary σ, such that the Gaifman graph of A does not contain H as a topological subgraph, and*
- *a successor-invariant formula $\varphi \in FO[\sigma_{\text{succ}}]$*

and checks whether

$$A \models \varphi$$

in time $f(|V(H)| + |\varphi|) \cdot |V(A)|^c$ for some computable function f and $c \in \mathbb{N}$, both depending only on $\mathbb{A}$.

Note that model-checking for first-order logic on nowhere dense classes of graphs is possible in time $f(|\varphi|) \cdot |V(A)|^{1+\epsilon}$ for arbitrarily small $\epsilon > 0$ by a result of Grohe et al. [21]. Even though a representation of a structure A in computer memory is likely to induce a linear order on the elements of $V(A)$, making this linear order or its successor relation accessible to the formula φ potentially complicates the model-checking problem. In particular, adding the cycle corresponding to this linear order (or any other cycle through the whole graph) to A may introduce new shallow minors.

The proof of Theorem 1 is based on the following two lemmas:

▶ **Lemma 2.** *For every finite graph H there are constants $k \in \mathbb{N}$ and $c \in \mathbb{N}$ such that for every graph G which does not contain H as a topological subgraph there is a graph G' and a k-walk $w : [\ell] \to V(G')$ through G' such that G' is obtained from G by only adding edges and G' does not contain K_c as a topological subgraph. Furthermore, k, c, G' and w can be computed, given G and H, in time $f(|V(H)|) \cdot |V(G)|^d$ for some computable function f and $d \in \mathbb{N}$.*

▶ **Lemma 3.** *Let σ be a finite relational vocabulary, A a finite σ-structure, and $w : [\ell] \to V(A)$ a k-walk through the Gaifman graph of A.*

Then there is a finite relational vocabulary σ_k and a first-order fomula $\varphi^{(k)}_{\mathrm{succ}}(x, y)$, both depending only on k, and a $(\sigma \cup \sigma_k)$-expansion A' of A which can be computed from A and w in polynomial time, such that

- *The Gaifman-graphs of A' and A are the same,*
- *$\varphi^{(k)}_{\mathrm{succ}}$ defines a successor relation on A'.*

Lemma 3 is taken from [13, Lemma 4.4] and has been proved there. We will prove Lemma 2 in Section 4. The proof of Theorem 1 then is a combination of the above lemmas:

Proof of Theorem 1. Given a σ-structure A, a successor-invariant σ_{succ}-formula φ and a graph H which is not a topological subgraph of the Gaifman graph of A, we first compute the Gaifman graph G of A. Using the algorithm of Lemma 2 we then compute a k-walk $w : [\ell] \to V(A)$ through a supergraph G' of G which excludes some clique $K_{c'}$ as a topological subgraph.

Let E be a binary relation symbol. We expand A to a $(\sigma \cup \{E\})$-structure A' by setting

$$E(A') := \{(w(i), w(i+1)) \mid i \in [\ell - 1]\} \cup \{(w(\ell), w(1))\}.$$

Then G' is the Gaifman graph of A', which by Lemma 2 excludes K_c as a topological subgraph.

Using Lemma 3 we compute, for a suitable $\tau \supseteq \sigma$, a τ-expansion A'' of A' and an FO[τ]-formula $\varphi^{(k)}_{\mathrm{succ}}(x, y)$ which defines a successor relation on A''. We replace all atomic subformulae $\operatorname{succ} xy$ in φ by $\varphi^{(k)}_{\mathrm{succ}}(x, y)$, obtaining an FO[$\tau$]-formula $\tilde{\varphi}$ such that

$$A'' \models \tilde{\varphi} \quad \Leftrightarrow \quad (A, S) \models \varphi$$

where S the successor relation defined by $\varphi^{(k)}_{\mathrm{succ}}$. Note $\varphi^{(k)}_{\mathrm{succ}}$ and τ depend only on k, which in turn only depends on H.

Since the Gaifman graph G'' of A'' excludes H as a topological subgraph, there is a class $\mathcal{C}$ of graphs of bounded expansion such that $G'' \in \mathcal{C}$. We can therefore use Dvořák et al.'s model-checking algorithm [9] for FO on $\mathcal{C}$ to check whether

$$A'' \models \tilde{\varphi}$$

in time linear in $|A|$. ◀

4 k-walks in Graphs with Excluded Topological Subgraphs

In this section we will prove Lemma 2. Given a graph G which excludes a graph H as a topological subgraph, as a first step towards constructing a supergraph G' with a k-walk we compute a tree-decomposition of G into graphs which exclude H as a minor and graphs of almost bounded degree:

▶ **Theorem 4** (Theorem 4.1 in [22]). *For every $k \in \mathbb{N}$ there exists a constant $c = c(k) \in \mathbb{N}$ such that the following holds: If H is a graph on k vertices and G a graph which does not contain H as a topological subgraph, then there is a tree-decomposition $(\mathcal{T}, \mathcal{V})$ of G of adhesion at most c such that for all $t \in \mathcal{T}$*

- *$\bar{\mathcal{V}}_t$ has at most c vertices of degree larger than c,* or
- *$\bar{\mathcal{V}}_t$ excludes K_c as a minor.*

Furthermore, there is an algorithm that, given graphs G of size n and H of size k computes such a decomposition in time $f(k) \cdot n^{O(1)}$ for some computable function $f : \mathbb{N} \to \mathbb{N}$.

For the rest of this section we assume a graph $G = (V, E)$ together with a tree-decomposition $(\mathcal{T}, \mathcal{V})$ satisfying the properties of Theorem 4 as given. We will construct k-walks through each of the bags of this decomposition, for a suitable k depending only on H, suitably adding edges within the bags in a way that will not create large topological subgraphs. We will then connect these k-walks to obtain a k'-walk through all of G, carefully adding further edges where necessary.

If $s, t \in \mathcal{T}$ are neighbours in $\mathcal{T}$ we will connect the k-walk through $\mathcal{V}_s$ and the k-walk through $\mathcal{V}_t$ by joining them along a suitably chosen vertex $v \in \mathcal{V}_s \cap \mathcal{V}_t$. Since the resulting walk may visit v a total of $k + 1$ times, we must be careful not to select the same vertex v more than a bounded number of times.

We first pick an arbitrary tree node $r \in \mathcal{T}$ as the root of the tree-decomposition. Notions such as parent and sibling nodes are meant with respect to this root node r. For a node $t \in \mathcal{T}$ we define its adhesion $\alpha_t \subseteq \mathcal{V}_t$ as

$$\alpha_t := \begin{cases} \emptyset & \text{if } t = r \\ \mathcal{V}_s \cap \mathcal{V}_t & \text{if } s \text{ is the parent of } t. \end{cases}$$

By adding the necessary edges within the bags we may assume that each $\mathcal{V}_t$ is identical to its torso, in other words we may assume that $G[\alpha_t]$ is a clique for each $t \in \mathcal{T}$.

4.1 Computing the k-walks w_t

Let $s, t \in \mathcal{T}$ be nodes such that s is the parent of t. It may happen that $\alpha_s \cap \alpha_t \neq \emptyset$, and in fact we can not bound

$$|\{s \in \mathcal{T} \mid v \in \mathcal{V}_s\}|$$

for all G excluding a fixed topological subgraph and all $v \in V(G)$. Since we are only allowed to visit each vertex a bounded (for a fixed excluded topological subgraph) number of times, we first compute, for $t \in \mathcal{T}$, a k-walk w_t through a suitable supergraph of $\mathcal{V}_t \setminus \alpha_t$.

If $\bar{\mathcal{V}}_t$ contains only c vertices of degree larger than c we choose an arbitrary enumeration $v_1, \ldots, v_\ell$ of $\mathcal{V}_t \setminus \alpha_t$ and add edges

$$v_1v_2, v_2v_3, \ldots, v_{\ell-1}v_\ell, v_\ell v_1$$

to G as far as they are not already present. This will increase the degree of each vertex by at most 2, so there are still at most c vertices of degree larger than $c+2$. We set

$$\begin{aligned} w_t : [\ell] &\to \mathcal{V}_t \\ i &\mapsto v_i \end{aligned}$$

for these bags.

If, on the other hand, $\bar{\mathcal{V}}_t$ exludes a clique K_c as a minor, we invoke the following lemma on the graph $\mathcal{V}_t \setminus \alpha_t$:

▶ **Lemma 5** (Lemma 3.3 in [13]). *For every natural number c there are $k, c' \in \mathbb{N}$ such that: If $G = (V, E)$ is a graph which does not contain a K_c-minor, then there is a supergraph $G' = (V, E')$ obtained from G by possibly adding edges such that G' does not contain a $K_{c'}$-minor and there is a k-walk w through G'. Moreover, G' and w can be found in polynomial time for fixed c.*

Since we ignore the vertices in α_t when computing the k-walk w_t, it may happen that the resulting supergraph of $\bar{\mathcal{V}}_t$ *does* contain a $K_{c'}$-minor. However, the largest possible clique minor is still of bounded size, because $|\alpha_t| \leq c$:

▶ **Lemma 6.** *Let $G = (V, E)$ be a graph such that $K_{c'} \not\preceq G$, and let $G \oplus K_c$ be the graph with vertex set $V' = V \cup [c]$ and edge set*

$$E' = E \cup \binom{[c]}{2} \cup \{va \mid v \in V, a \in [c]\}.$$

In other words, $G \oplus K_c$ is the disjoint sum of G and K_c plus edges between all vertices of G and all vertices of K_c. Then $K_{c+c'} \not\preceq G \oplus K_c$.

Proof. Otherwise let $X_1, \ldots, X_{c+c'}$ be the branch sets of a $K_{c+c'}$-minor in $G \oplus K_c$. At most c of the sets contain vertices of the added K_c-clique. The remaining sets form the branch sets of a K_c-minor in G, contradicting the assumption that $K_c \not\preceq G$. ◀

4.2 Connecting the k-walks

We still need to connect the k-walks through the individual bags of $(\mathcal{T}, \mathcal{V})$ to obtain a single k'-walk through the whole graph, for some k' to be determined below. This is the most complicated part of our construction, since we must guarantee that no vertex is visited more than k' times by the resulting walk, and that no large topological clique subgraphs are created.

In the case of graphs excluding some fixed minor, the Graph Structure Theorem guarantees the existence of a tree-decomposition into nearly embeddable graphs such that neighbouring bags intersect only in apices and vertices lying on some face or vortex of their near embeddings, and this was used in [13] to select vertices from the adhesion sets of bags in a suitable way. Since the decomposition theorem for graphs excluding a topological minor does not provide this kind of information, we need a different approach here. Instead, our method for selecting vertices along which to connect the k-walks relies on the fact that sparse graphs are *degenerate*, i.e. every subgraph of a sparse graph contains some vertex of small degree.

In connecting the walks w_t, we will proceed down the tree $\mathcal{T}$. At any point in the process we keep a set $D \subseteq \mathcal{T}$ and a walk w such that

- D is a connected subset of $\mathcal{T}$,
- the k'-walk has been constructed in $\bigcup_{t \in D} \mathcal{V}_t$,

- if $s \in D$ and s' is a sibling of s then also $s' \in D$,
- w is a k'-walk through $\bigcup_{t \in D} \mathcal{V}_t$, and if $s \in D$ has a child $t \notin D$, then the vertices in $\mathcal{V}_s \setminus \alpha_s$ are visited at most $k+1$ times by w.

We start with $D = \{r\}$ and $w = w_r$, where r is the root of $\mathcal{T}$. This is easily seen to satisfy all of the above conditions.

Now let $s \in D$ be a node whose children $t_1, \ldots, t_n$ are not in D. We let

$$C_i := \alpha_{t_i} \setminus \alpha_s$$

be the adhesion set of t_i with all vertices of the adhesion set of s removed. If $C_i = \emptyset$ then t_i can be made a sibling of s (rather than a child), so we assume that all C_i are nonempty. Since the properties of $(\mathcal{T}, \mathcal{V})$ are guaranteed for the torsos of the bags we may assume that $G[C_i]$ is a clique for each i and that w visits the vertices of $\bigcup C_i$ at most $k+1$ times.

It may happen that $C_i = C_j$ for some $i \neq j$. To deal with this, assume that

$$C_1 = C_2 = \cdots = C_m \neq C_i \text{ for i} > \text{m}.$$

For each $i = 1, \ldots, m$ we choose an edge $u_i v_i \in E(\mathcal{V}_{t_i})$ which is traversed by the walk w_{t_i} in the direction from u_i to v_i at some point. We add edges

$$u_i v_{i+1} \text{ for } i = 1, \ldots, m-1 \text{ and } u_m v_1$$

and connect the walks $w_{t_1}, \ldots, w_{t_m}$ along these edges. Because w_{t_i} is a walk through $\mathcal{V}_{t_i} \setminus \alpha_{t_i}$, we have

$$u_i, v_i \in \mathcal{V}_{t_j} \quad \Leftrightarrow \quad i = j$$

for $i, j = 1, \ldots, m$. To accomodate for the extra edges, we add the vertices u_i and v_i to $\mathcal{V}_s$, and therefore to α_{t_i} and C_i. Since these vertices together with the added edges form an isolated cycle

$$u_1 v_1 u_2 v_2 \ldots u_m v_m u_1$$

in $\mathcal{V}_s$, no new topological subgraphs are created by this. The maximal adhesion of $(\mathcal{T}, \mathcal{V})$ is still bounded by $c+2$.

Therefore we now assume that the cliques $C_1, \ldots, C_n$ are all distinct. It remains to find a function

$$f : [n] \to V$$

such that

- $f(i) \in C_i$ for all i, and
- $\left|f^{-1}(v)\right| \leq M$ for all $v \in V$ and some constant M depending only on H.

We define the function f iteratively on larger subsets of $[n]$ as follows: Let $\tilde{G}$ be the subgraph of G induced on the union of all C_i:

$$\tilde{G} = G\left[\bigcup_i C_i\right].$$

We show that $\tilde{G}$ contains a vertex of degree (in $\tilde{G}$) at most d, for some constant d depending only on the constant c from Theorem 4 (and therefore only on the excluded topological subgraph H we started with). If $\mathcal{V}_s$ contains only c vertices of degree larger than c then

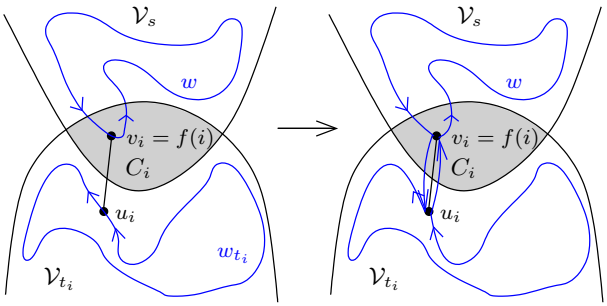


Figure 2 Connecting the individual k-walks.

this is true with $d = c$. If $\mathcal{V}_t$ excludes some clique K_c as a minor we use the fact that these graphs are d-degenerate for some d depending only on c. In fact, by Theorem 7.2.1 in [7] there is a constant d such that if the average degree of $\tilde{G}$ is at least d, then $K_c \preceq_{\text{top}} \tilde{G}$ and therefore $K_c \preceq \tilde{G}$.

In both cases there is a $v \in \bigcup_i C_i$ which has degree at most d in $\tilde{G}$. We want to bound the number of $i \in [n]$ for which $v \in C_i$. Since every clique C_i has size at most $c+2$, and if $v \in C_i$ then all elements of $C \setminus \{v\}$ are neighbours of v, there can be at most

$$M := \binom{d}{0} + \binom{d}{1} + \cdots + \binom{d}{c+1}$$

many such C_i, and this bound only depends on c. It is therefore safe to define

$$f(i) := v \text{ for all } i \in [n] \text{ such that } v \in C_i.$$

We remove these cliques and iterate until no cliques remain.

Once the function f has been found we connect the walk w through $\bigcup_{t \in D} \mathcal{V}_t$ with the walks w_{t_i} through the bags V_{t_i}. Let $w : [\ell] \to V$ be the walk constructed so far. For each $i \in [n]$ let $v_i = f(i) \in C_i$ be the vertex chosen by f, and let $u_i \in \mathcal{V}_{t_i} \setminus \alpha_{t_i}$ be a neighbour of v_i. If no such neighbour exists it is safe to create one by adding an edge between v_i and an arbitrary vertex of $\mathcal{V}_{t_i} \setminus \alpha_{t_i}$. We now extend the walk w by inserting the k-walk w_{t_i} along the edge $v_i u_i$ when v_i is first visited by w. This increases the number of times v_i and u_i are visited by one each (cf. Figure 2).

After inserting all walks $w_{t_1}, \ldots, w_{t_n}$ we set

$$D := D \cup \{t_1, \ldots, t_n\}$$

and repeat the process until $D = \mathcal{T}$. Note that the resulting walk is a $(k + M + 1)$-walk through the supergraph G' of G obtained by adding edges to G.

4.3 Topological Subgraphs in G'

By now we have a supergraph G' of G, obtained by only adding edges, and a $k' = (k+M+1)$-walk $w : [\ell] \to V(G')$ through this supergraph. Furthermore, there is a $c' = c'(H)$ depending only on (the size of) H and a tree-decomposition $(\mathcal{T}, \mathcal{V})$ of G' such that if $s, t \in \mathcal{T}$ then $|\mathcal{V}_s \cap \mathcal{V}_t| \leq c'$ and for all $t \in \mathcal{T}$

- $\bar{\mathcal{V}}_t$ has at most c' vertices of degree larger than c' or
- $\bar{\mathcal{V}}_t$ excludes $K_{c'}$ as a minor.

We show that this implies $K_{c'+2} \not\preceq_{\text{top}} G'$: Assume for a contradiction that $K_{c'+2} \preceq_{\text{top}} G$, and let $v_1, \ldots, v_{c'+2}$ be the branch vertices of a $K_{c'+2}$-subdivision in G. Then there is a $t \in \mathcal{T}$ such that $\{v_1, \ldots, v_{c'+2}\} \subseteq \mathcal{V}_t$: Otherwise choose $i < j$ and $t \neq t'$ so that

$$v_i \in \mathcal{V}_t \setminus \mathcal{V}_{t'} \quad \text{and} \quad v_j \in \mathcal{V}_{t'} \setminus \mathcal{V}_t.$$

Then, since the adhesion of $(\mathcal{T}, \mathcal{V})$ is at most c', there is a set $S \subseteq V$ of size at most c' separating two branch vertices, which is not possible in a $(c'+2)$-clique.

Now let $t \in \mathcal{T}$ be a tree node for which $\mathcal{V}_t$ contains all branch vertices. For $i < j$, let P_{ij} be the path in G connecting v_i and v_j. If all vertices on this path are in $\mathcal{V}_t$ we are done. Otherwise we may shorten this path to get a path P'_{ij} connecting v_i and v_j in the torso of $\mathcal{V}_t$. Thus

$$K_{c'+2} \preceq_{\text{top}} \mathcal{V}_t.$$

But none of the bags $\mathcal{V}_t$ can contain $K_{c'+2}$ as a topological subgraph: Since $K_{c'+2} \preceq_{\text{top}} \mathcal{V}_t$ implies $K_{c'+2} \preceq \mathcal{V}_t$ which in turn implies $K_{c'} \preceq \mathcal{V}_t$, none of the bags excluding $K_{c'}$ as a minor can contain $K_{c'+2}$ as a topological subgraph. But if $K_{c'+2} \preceq_{\text{top}} \mathcal{V}_t$ then there must be at least $c'+2$ vertices of degree at least $c'+1$, namely the branch vertices of the image of a subdivision of $K_{c'+2}$. We conclude that $K_{c'+2} \not\preceq_{\text{top}} G'$.

5 Dense Graphs

While model-checking for first-order logic has been studied rather thoroughly for sparse graph classes, few results are known for dense graphs:

- On classes of graphs with bounded clique-width (or, equivalently, bounded rank-width; cf. [26]), model-checking even for monadic second-order logic has been shown to be fpt by Courcelle et al. [5].
- More recently, model-checking on coloured posets of bounded width has been shown to be in fpt for existential FO by Bova et al. [3] and for all of FO by Gajarský et al. [19].

Both of these results extend to order-invariant FO, and therefore also to successor-invariant FO. For bounded clique-width, this has already been shown by Engelmann et al. in [14, Thm. 4.2]. For posets of bounded width we give a proof here. We first review the necessary definitions:

▶ **Definition 7.** A *partially ordered set (poset)* $(P, \leq^P)$ is a set P with a reflexive, transitive and antisymmetric binary relation $\leq^P$. A *chain* $C \subseteq P$ is a totally ordered subset, i.e. for all $x, y \in C$ one of $x \leq^P y$ and $y \leq^P x$ holds. An *antichain* is a set $A \subseteq P$ such that if $x \leq^P y$ for $x, y \in A$ then $x = y$. The *width* of $(P, \leq^P)$ is the maximal size $|A|$ of an antichain $A \subseteq P$. A *coloured* poset is a poset $(P, \leq^P)$ together with a function $\lambda : P \to \Lambda$ mapping P to some set Λ of *colours*. By $\|P\|$ we denote the length of a suitable encoding of $(P, \leq^P)$.

We will need Dilworth's Theorem, which relates the width of a poset to the minimum number of chains needed to cover the poset:

▶ **Theorem 8** (Dilworth's Theorem). *Let $(P, \leq^P)$ be a poset. Then the width of $(P, \leq^P)$ is equal to the minimum number k of disjoint chains $C_i, \ldots, C_k \subseteq P$ needed to cover P, i.e. such that $\bigcup_i C_i = P$.*

A proof can be found in [7, Sec. 2.5]. Moreover, by a result of Felsner et al. [15], both the width w and a set of chains $C_1, \ldots, C_w$ covering P can be computed from $(P, \leq_P)$ in time $O(w \cdot \|P\|)$.

With this, we are ready to prove the following:

▶ **Theorem 9.** *There is an algorithm which, on input a coloured poset $(P, \leq^P)$ with colouring $\lambda : P \to \Lambda$ and an order-invariant first-order formula φ, checks whether $P \models \varphi$ in time $f(w, |\varphi|) \cdot \|P\|^2$ where w is the width of $(P, \leq^P)$.*

Proof. Using the algorithm of [15], we compute a chain cover $C_1, \dots, C_w$ of $(P, \leq^P)$. To obtain a linear order on P, we just need to arrange the chains in a suitable order, which can be done by colouring the vertices with colours $\Lambda \times [w]$ via

$$\lambda'(v) = (\lambda(v), j) \text{ for } v \in C_j.$$

Then

$$\varphi_{\leq}(x,y) := \Bigg(\bigvee_{\substack{\lambda_x, \lambda_y \in \Lambda, \\ i<j}} (\lambda'(x) = (\lambda_x, i) \wedge \lambda'(y) = (\lambda_y, j)) \Bigg) \vee \Bigg(\bigvee_{\substack{\lambda_x, \lambda_y \in \Lambda, \\ i \in [w]}} \lambda'(x) = (\lambda_x, i) \wedge \lambda'(y) = (\lambda_y, i) \wedge x \leq y \Bigg)$$

defines a linear order on $(P, \leq_P)$ with colouring λ'. After substituting $\varphi_{\leq}$ for $\leq$ in φ we may apply Gajarský et al.'s algorithm [19] to check whether $P \models \varphi$. ◀

6 Conclusion and Further Research

We have shown that model-checking for successor-invariant first-order logic is fixed-parameter tractable on classes of graphs excluding some fixed graph H as a topological subgraph. This extends previous results showing tractibility on planar graphs [14] and graphs with excluded minors [13]. For dense graphs, we showed how the recent model-checking algorithm by Gajarský et al. [19] can be adapted to order-invariant FO.

This prompts for further generalisation in two ways: First, can we close the gap between plain first-order logic and its successor-invariant counterpart? Next steps could be graph classes with bounded expansion or with locally excluded minors. However, no structure theorem comparable to those of Robertson and Seymour and of Grohe and Marx are known for these graph classes.

Another interesting open question is whether model-checking for order-invariant first-order logic is tractable on any of the classes depicted in Figure 1. Since the Gaifman graph of a linearly ordered structure is a clique, there is no hope of finding a "good" linear order which can be added to the input structure without destroying the desirable properties of its Gaifman graph. As shown in [23], order-invariant first-order logic has a Gaifman-style locality property (see also [1]). It is, however, not at all clear how this could be turned into an efficient model-checking algorithm. In particular, no variant of Gaifman normal form is known for this logic.

References

1 Matthew Anderson, Dieter van Melkebeek, Nicole Schweikardt, and Luc Segoufin. Locality from circuit lower bounds. *SIAM J. Comput.*, 41(6):1481–1523, 2012.

2 Michael Benedikt and Luc Segoufin. Towards a characterization of order-invariant queries over tame graphs. *J. Symb. Log.*, 74(1):168–186, 2009.

3 Simone Bova, Robert Ganian, and Stefan Szeider. Model checking existential logic on partially ordered sets. *ACM Transactions on Computational Logic (TOCL)*, 17(2):10, 2015.

4 Bruno 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.

5 Bruno Courcelle, Johann Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

6 Anuj Dawar, Martin Grohe, and Stephan Kreutzer. Locally excluding a minor. In *Logic in Computer Science (LICS)*, pages 270–279, 2007.

7 Reinhard Diestel. *Graph Theory*. Number 173 in GTM. Springer, 4th edition, 2012.

8 Rod Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1998.

9 Zdeněk Dvořák, Daniel Král, and Robin Thomas. Testing first-order properties for subclasses of sparse graphs. *Journal of the ACM (JACM)*, 60(5):36, 2013.

10 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer, 2nd edition, 1999.

11 Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Springer, 2nd edition, 1994.

12 Kord Eickmeyer, Michael Elberfeld, and Frederik Harwath. Expressivity and succinctness of order-invariant logics on depth-bounded structures. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, pages 256–266, 2014.

13 Kord Eickmeyer, Ken-Ichi Kawarabayashi, and Stephan Kreutzer. Model checking for successor-invariant first-order logic on minor-closed graph classes. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 134–142. IEEE Computer Society, 2013.

14 Viktor Engelmann, Stephan Kreutzer, and Sebastian Siebertz. First-order and monadic second-order model-checking on ordered structures. In *Logics in Computer Science*, pages 275–284, 2012.

15 Stefan Felsner, Vijay Raghavan, and Jeremy Spinrad. Recognition algorithms for orders of small width and graphs of small dilworth number. *Order*, 20(4):351–364, 2003.

16 Jörg Flum and Martin Grohe. Fixed-parameter tractability, definability, and model-checking. *SIAM J. Comput.*, 31(1):113–145, 2001.

17 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. ISBN 3-54-029952-1.

18 Markus Frick and Martin Grohe. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM*, 48:1148–1206, 2001.

19 Jakub Gajarský, Petr Hliněný, Daniel Lokshtanov, Jan Obdržálek, Sebastian Ordyniak, MS Ramanujan, and Saket Saurabh. Fo model checking on posets of bounded width. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 963–974. IEEE, 2015.

20 Martin Grohe. Logic, graphs, and algorithms. In E. Grädel, T. Wilke, and J. Flum, editors, *Logic and Automata – History and Perspectives*. Amsterdam University Press, 2007.

21 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding first-order properties of nowhere dense graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 89–98. ACM, 2014.

22 Martin Grohe and Dániel Marx. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM Journal on Computing*, 44(1):114–159, 2015.

23 Martin Grohe and Thomas Schwentick. Locality of order-invariant first-order formulas. *ACM Trans. Comput. Logic*, 1(1):112–130, July 2000.

24 Stephan Kreutzer. Algorithmic meta-theorems. In Javier Esparza, Christian Michaux, and Charles Steinhorn, editors, *Finite and Algorithmic Model Theory*, London Mathematical Society Lecture Note Series, chapter 5, pages 177–270. Cambridge University Press, 2011.

a preliminary version is available at Electronic Colloquium on Computational Complexity (ECCC), TR09-147, `http://www.eccc.uni-trier.de/report/2009/147`.

25 Leonid Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. Spinger-Verlag, 2004.

26 Sang-Il Oum and Paul D. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96:514–528, 2006.

27 Benjamin Rossman. Successor-invariant first-order logic on finite structures. *J. Symb. Log.*, 72(2):601–618, 2007.

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

29 William T. Tutte. *Graph Theory*, volume 21 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2001.

Definability of Cai-Fürer-Immerman Problems in Choiceless Polynomial Time

Wied Pakusa[1], Svenja Schalthöfer[2], and Erkal Selman[3]

1 **Mathematical Foundations of Computer Science, RWTH Aachen University, Germany**
pakusa@logic.rwth-aachen.de
2 **Mathematical Foundations of Computer Science, RWTH Aachen University, Germany**
schalthoefer@logic.rwth-aachen.de
3 **Logic and Theory of Discrete Systems, RWTH Aachen University, Germany**
selman@informatik.rwth-aachen.de

Abstract

Choiceless Polynomial Time (CPT) is one of the most promising candidates in the search for a logic capturing Ptime. The question whether there is a logic that expresses exactly the polynomial-time computable properties of finite structures, which has been open for more than 30 years, is one of the most important and challenging problems in finite model theory.

The strength of Choiceless Polynomial Time is its ability to perform isomorphism-invariant *computations* over structures, using hereditarily finite sets as data structures. But, as it preserves symmetries, it is choiceless in the sense that it cannot select an arbitrary element of a set—an operation which is crucial for many classical algorithms. CPT can define many interesting Ptime queries, including (the original version of) the Cai-Fürer-Immerman (CFI) query. The CFI query is particularly interesting because it separates fixed-point logic with counting from Ptime, and has since remained the main benchmark for the expressibility of logics within Ptime.

The CFI construction associates with each connected graph a set of CFI-graphs that can be partitioned into exactly two isomorphism classes called odd and even CFI-graphs. The problem is to decide, given a CFI-graph, whether it is odd or even. In the original version, the underlying graphs are linearly ordered, and for this case, Dawar, Richerby and Rossman proved that the CFI query is CPT-definable. However, the CFI query over general graphs remains one of the few known examples for which CPT-definability is open.

Our first contribution generalises the result by Dawar, Richerby and Rossman to the variant of the CFI query where the underlying graphs have colour classes of logarithmic size, instead of colour class size one. Secondly, we consider the CFI query over graph classes where the maximal degree is linear in the size of the graphs. For these classes, we establish CPT-definability using only sets of small, constant rank, which is known to be impossible for the general case.

In our CFI-recognising procedures we strongly make use of the ability of CPT to create sets, rather than tuples only, and we further prove that, if CPT worked over tuples instead, no such procedure would be definable. We introduce a notion of "sequence-like objects" based on the structure of the graphs' symmetry groups, and we show that no CPT-program which only uses sequence-like objects can decide the CFI query over complete graphs, which have linear maximal degree. From a broader perspective, this generalises a result by Blass, Gurevich, and van den Bussche about the power of isomorphism-invariant machine models (for polynomial time) to a setting with counting.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases finite model theory, descriptive complexity, logic for Ptime, Choiceless Polynomial Time, Cai-Fürer-Immerman

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.19

© Wied Pakusa, Svenja Schalthöfer, and Erkal Selman;
licensed under Creative Commons License CC-BY

25th EACSL Annual Conference on Computer Science Logic (CSL 2016).
Editors: Jean-Marc Talbot and Laurent Regnier; Article No. 19; pp. 19:1–19:17

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

1 Introduction

One of the most important questions in descriptive complexity theory is whether there is a logic capturing Ptime [7, 14, 15].

Currently, there are two main branches of research approaching that problem. The first sets off from a seminal result of Immerman and Vardi which shows that (least) fixed-point logic captures Ptime on *ordered* structures [16, 19]. This precise logical characterisation of polynomial time heavily relies on the presence of a linear order on the input structure: Unfortunately, there are many queries, including trivial counting properties, which cannot be defined in fixed-point logic without a linear order. Still, the Immerman-Vardi Theorem indicates that fixed-point logic is a reasonable starting point to search for a logic capturing polynomial time. Indeed, much research has focused on ways to extend fixed-point logic by new operators in order to capture larger and larger fragments of Ptime "from below".

The best-studied of such formalisms is the extension of fixed-point logic by a counting mechanism, called fixed-point logic with counting or FP+C. Immerman proposed FP+C as a candidate to capture polynomial time on all finite structures. Indeed, it turned out that FP+C is very powerful and can express a robust and natural fragment of polynomial time. For a recent survey on FP+C, see [8].

However, Cai, Fürer and Immerman [6] found a very clever construction which demonstrates that FP+C fails to capture Ptime on all structures. This Cai-Fürer-Immerman query has since served as an extremely valuable tool for analysing the expressive power of logics over finite structures, as well as the complexity of the graph isomorphism problem.

Besides FP+C, further extensions of fixed-point logic by much stronger operators have been studied in recent years. The most important such extension is Rank Logic, proposed by Dawar, Grohe, Holm and Laubner in [9]. It is still open whether certain variants of Rank Logic [13] capture polynomial time.

Whereas studying extensions of fixed-point logic can be seen as searching for a logic for Ptime "from below", the second approach attacks the problem "from above". That branch of research aims for a formalism of Ptime *computation* that operates on structures and is *isomorphism-invariant*. Indeed, isomorphism-invariance is the main difference between logics and machine models, according to the standard definition, due to Gurevich [15], of what would constitute a logic for Ptime. Choiceless Polynomial Time, introduced by Blass, Gurevich and Shelah [3], approaches Ptime in that sense, with the intuition that it resembles classical Ptime computation "as closely as possible" without violating isomorphism-invariance.

Choiceless Polynomial Time

As it is isomorphism-invariant, Choiceless Polynomial Time, in contrast to actual computation models, lacks arbitrary choice: instructions like "choose an arbitrary vertex" are not possible. Instead, CPT performs parallel computations on possibly nested, logically definable sets over domain elements, using ordinals for counting operations. More precisely, CPT is able to construct hereditarily finite sets over the input structure, allowing for higher-order data structures. Because of its rather algorithmic nature, we often refer to CPT-programs or -algorithms instead of CPT-formulas. Most characteristics of a logic capturing Ptime are rather easy to verify for CPT. The challenge is to determine whether it can express *all* Ptime queries. To date, CPT has not been separated from Ptime, and many queries proposed for that purpose have been shown to be expressible, such as:

- Any query definable in FP+C is also definable in CPT.
- On structures with sufficiently large padding, CPT captures PTIME [4, 17]. This already implies that CPT is a strict extension of FP+C.
- The Cai-Fürer-Immerman query is definable in CPT, even without counting, if the underlying graphs are linearly ordered [10].
- CPT captures PTIME on structures with bounded colour class size if the colour classes have Abelian automorphism groups. A subprocedure shows that solvability of *cyclic linear equation systems* is CPT-definable [1].

For a more detailed survey of some recent insights, see also [11].

These results illustrate the surprising expressive power of Choiceless Polynomial Time. However, there are some properties for which we do not know whether they can be expressed in CPT, and which might witness a separation from polynomial time.

Most importantly, it is open whether the Cai-Fürer-Immerman problem over general graphs can be expressed in Choiceless Polynomial Time. Since the CFI query has proven to be an extremely valuable benchmark for the expressive power of logics over finite structures, solving this question would significantly increase our understanding of the expressive power of CPT. The CFI query additionally gives rise to particularly simple instances of various more general, and arguably more important, queries for which we do not know whether they can be expressed in Choiceless Polynomial Time. In particular, it is open whether CPT can define the isomorphism problem for graphs of bounded degree or solve linear equation systems over finite fields. For both of these problems certain variants of the CFI query constitute a rather simple class of instances, which means that it should be easier to find a CPT-procedure for the CFI query first before trying to solve these much more sophisticated problems in CPT in general.

The Cai-Fürer-Immerman Query

The CFI query is a subproblem of the graph isomorphism problem. With each *connected* graph $G = (V, E)$, the CFI construction associates a set of CFI-graphs $\mathfrak{G}^T$ over G, for every $T \subseteq V$, by replacing each vertex v of G by a certain graph gadget. More precisely, each vertex $v \in V$ can either be replaced by an *even* gadget ($v \notin T$) or by an *odd* gadget ($v \in T$). Hence, the CFI construction associates with every graph G an exponential-sized set of CFI-graphs $\{\mathfrak{G}^T : T \subseteq V\}$. However, it turns out that, up to isomorphism, there are in fact only two different CFI-graphs for fixed G. Indeed, a pair of CFI-graphs $\mathfrak{G}^T$ and $\mathfrak{G}^S$ over G is isomorphic if, and only if, the parity of the number of odd gadgets is the same, i.e. if $|T| \equiv |S| \pmod 2$. The CFI query is to determine, given a CFI-graph $\mathfrak{G}^T$, the parity of $|T|$. We give a precise definition of the CFI construction in Section 3.

It is known that the CFI query is decidable in PTIME, but not definable in FP+C [6]. Furthermore, it is definable in Rank Logic [9] and in Choiceless Polynomial Time without counting in case the underlying graph is *linearly ordered*, but not definable in CPT using only sets of bounded rank [10].

In the original work, Cai, Fürer and Immerman applied their construction to *ordered, three-regular* graphs. The effect is that the resulting CFI-graphs are also three-regular, and have colour class size four. Hence, Cai, Fürer and Immerman not only demonstrated that FP+C fails to capture polynomial time, but also that it cannot decide the isomorphism problem for graphs with bounded degree *and* bounded colour class size, though there are efficient isomorphism tests for both classes.

The immediate question is whether Choiceless Polynomial Time, as an extension of FP+C, can define these isomorphism problems and, in particular, whether it can distinguish

between the odd and even version of CFI-graphs. The first positive result in this context was achieved by Blass, Gurevich and Shelah [4], who proved that the Cai-Fürer-Immerman query can be expressed in CPT if the graphs come with a suitable padding, which makes it possible to define all linear orders on the input structure in spite of the space bound. This observation also lead to the separation of CPT and FP+C. However, it remained a challenging open question whether CPT can also define the CFI query without padding.

By using a very elegant construction, Dawar, Richerby and Rossman [10] were able to confirm this later for the special case where the underlying graphs are linearly ordered (as it is the case in the original construction of Cai, Fürer and Immerman). Essentially, the approach of Dawar, Richerby and Rossman is to succinctly represent the complete isomorphism class of a given CFI-graph—a class of exponential size—as a sufficiently small hereditarily finite set over the input structure. To this end they invent a data structure which uses highly nested sets. They further prove that this nesting cannot be avoided: with sets of constant rank it is not possible to define the CFI query over ordered graphs in CPT. Having a representation of the isomorphism class as a single object, they then show how to obtain, in CPT, a canonical numerical invariant for the given isomorphism class from which one can read off the parity of the CFI-graph. A more detailed description of this algorithm is given in Section 4.

In this paper, we set out to identify new classes of graphs over which the CFI query is CPT-definable. Moreover, we analyse the resources which are required by CPT-programs to decide the CFI query. Our first contribution is to generalise the result of Dawar, Richerby and Rossman from linearly ordered graphs to graphs with colour classes of *logarithmic* size, that is graphs with a built-in linear preorder on the universe which may contain classes of incomparable elements, but where the size of such classes is bounded logarithmically in the size of the input structure. This allows us to use all subsets of every colour class in a CPT-program. Note that the case of ordered graphs appears as the special case of colour classes of size one. Our procedure is based on the insight that it is possible to represent all linear orders which are consistent with the given preorder using polynomial resources if we iteratively join orderings defined over the same *set* of elements. The proof can be found in Section 5.

Secondly, we strengthen a claim from [10] for complete graphs and show that the CFI query over classes of *unordered* graphs where the maximal degree is linear is CPT-definable using only sets of constant rank. Recall that this is not possible in the general case.

If the underlying graph over n vertices has at least one vertex of degree $\frac{n}{k}$, the associated CFI-graph is of size $\geq 2^{\frac{n}{k}}$. Hence, a CPT-program can access *all subsets* of the vertex set of the underlying graph within the polynomial resource bounds. We use these subsets to inductively represent partial computations for the parity of the CFI-graph. Our construction is presented in Section 6.

Note that we *cannot* access all $n!$ different linear orderings of the underlying graphs with any resource bound polynomial in 2^n. Intuitively, this means that the power to use *sets* as abstractions of the linear orderings defined on their domain is vital for our CPT-procedure to respect polynomial bounds.

In Section 7 we use this observation to answer the following question: are isomorphism-invariant computation models with polynomial time bounds strictly more powerful if they use sets instead of only tuples? In the absence of counting, this question was answered by Blass, Gurevich and van den Bussche in [5]: set-like data structures are more powerful than sequence-like data-structures (see also [2].) We generalise this result to the case of counting and show that CPT-programs which decide the CFI query over complete graphs have to use set-like objects. In particular, this yields an interesting lower bound for a fragment of

Interpretation Logic, a characterisation of Choiceless Polynomial Time which was presented in [12]: Interpretation Logic without congruences cannot decide the Cai-Fürer-Immerman query over complete graphs, and is thus strictly less expressive than the fragment of CPT using only sets of bounded rank.

2 Choiceless Polynomial Time

Choiceless Polynomial Time is a polynomial time restriction of BGS logic (named after Blass, Gurevich and Shelah). The definition used here is based on the concise definition given by Rossman [18]. Let τ be a relational signature and let $\mathfrak{A} = (A, \tau)$ be a finite τ-structure. BGS logic is evaluated over the *hereditarily finite expansion* $\mathrm{HF}(\mathfrak{A})$ of $\mathfrak{A}$. The signature of $\mathrm{HF}(\mathfrak{A})$ is $\tau^{\mathrm{HF}} = \tau \uplus \{\emptyset, \mathrm{Atoms}, \in, \mathrm{Pair}, \mathrm{Union}, \mathrm{Unique}, \mathrm{Card}\}$, where $\emptyset$ and Atoms are constants, $\in$ is a binary relation symbol, Union, Unique and Card are unary function symbols and Pair is a binary function symbol. The domain of $\mathrm{HF}(\mathfrak{A})$ is the set of hereditarily finite sets over the *atoms* A. The *hereditarily finite sets* $\mathrm{HF}(A)$ are defined recursively as the finite sets of atoms and all finite sets of hereditarily finite objects, where an *object* is an atom or a set. In $\mathrm{HF}(\mathfrak{A})$, the relations and functions are interpreted as follows: $\emptyset^{\mathrm{HF}(\mathfrak{A})} = \emptyset$, $\mathrm{Atoms}^{\mathrm{HF}(\mathfrak{A})} = A$, $\in^{\mathrm{HF}(\mathfrak{A})} = \{(a, b) \mid a \in b\}$, $\mathrm{Pair}^{\mathrm{HF}(\mathfrak{A})}(a, b) = \{a, b\}$, $\mathrm{Union}^{\mathrm{HF}(\mathfrak{A})}(a) = \{b \in c : c \in a\}$, $\mathrm{Unique}^{\mathrm{HF}(\mathfrak{A})}(a) = b$ if $a = \{b\}$ for some b and $\emptyset$ otherwise, and $\mathrm{Card}^{\mathrm{HF}(\mathfrak{A})}(a) = |a|$ encoded as a von Neumann ordinal if a is a set and $\emptyset$ if $a \in A$.

BGS logic is defined by means of *terms*, *formulas* and *programs*. Terms are interpreted by values in $\mathrm{HF}(A)$. Syntactically, terms are variables, constants from τ^{HF}, objects $f\bar{t}$ for a function symbol $f \in \tau^{\mathrm{HF}}$ and terms $\bar{t}$, and *comprehension terms*. A comprehension term is of the form $t = \{s(\bar{x}, y) : y \in r(\bar{x}) : \varphi(\bar{x}, y)\}$, where s and r are terms and φ is a formula, with the semantics $\{s^{\mathfrak{A}}(\bar{a}, b) : y \in r^{\mathfrak{A}}(\bar{a}) : \mathfrak{A} \models \varphi(\bar{a}, b)\}$ for an assignment $\bar{a}$ of $\bar{x}$. A *formula* is either $t_1 = t_2$ or $R\bar{t}$ for terms $\bar{t} = t_1 \dots t_r$ and r-ary $R \in \tau$, or constructed from formulas with the connectives $\wedge, \vee, \neg$. A BGS *program* is a triple $\Pi = (\Pi_{\mathrm{step}}, \Pi_{\mathrm{halt}}, \Pi_{\mathrm{out}})$ of a term Π_{step} and formulas $\Pi_{\mathrm{halt}}, \Pi_{\mathrm{out}}$. The *run* of Π on $\mathfrak{A}$ is the sequence $(a_i)_{i \geq 0}$ such that $a_0 = \emptyset$ and $a_{i+1} = \Pi_{\mathrm{step}}(a_i)$ for $i > 0$. Choose $\kappa \leq \omega$ maximal such that $\mathfrak{A} \not\models \Pi_{\mathrm{halt}}(a_i)$ for $i < \kappa$. If κ is finite, then $\mathfrak{A} \models \Pi$ if and only if $a_\kappa \models \Pi_{\mathrm{out}}$.

To obtain Choiceless Polynomial Time, a fragment of BGS contained in PTIME, the complexity of the run and the occurring sets, measured in terms of their transitive closure, is bounded polynomially. A set y is *transitive* if $x \subseteq y$ for each $x \in y$. The *transitive closure* $\mathrm{tc}(x)$ of a set x is the least transitive set y with $x \subseteq y$. A CPT *program* is a pair $\bar{\Pi} = (\Pi, p)$, where Π is a BGS program and $p : \mathbb{N} \to \mathbb{N}$ a polynomial. The run of $\bar{\Pi}$ on $\mathfrak{A}$ is the maximal initial segment ρ of the run of Π on $\mathfrak{A}$ such that the length of ρ is at most $p(|A|)$ and, for each a_i in ρ, $\mathrm{tc}(a_i) \leq p(|A|)$. By CPT, we denote the variant with the function Card defined previously, unless stated otherwise.

We extend the syntax of CPT by assuming that there are distinct constants $0, 1$, and that there are syntactic means to define *tuples* explicitly. Note that tuples can be encoded as an extension of the Kuratowski encoding of pairs, so this is indeed only a change of syntax. We denote tuples as objects $\langle a_1, \dots, a_k \rangle$.

3 Graphs and the Cai-Fürer-Immerman Construction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. For a vertex $v \in V$, $E(v)$ denotes the set of edges incident to v. If $V', V'' \subseteq V$, $E[V']$ is the set of edges in the subgraph induced by V', and (V', V'') denotes the (V', V'')-cut, i.e. all edges $\{v, w\}$ with $v \in V'$ and $w \in V''$.

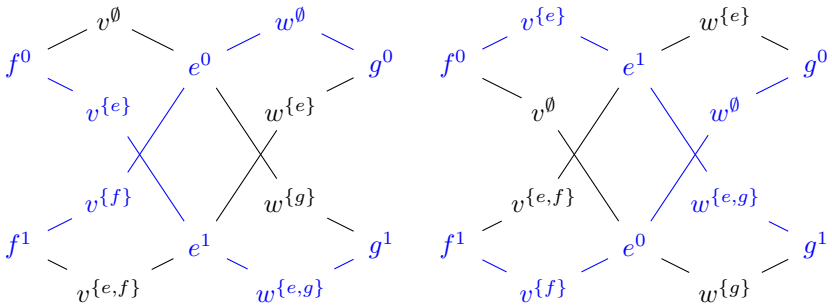


Figure 1 A subgraph of the graph $\mathfrak{G}$. The subgraph highlighted on the left corresponds to the CFI-graph $\mathfrak{G}^{\{v\}}$. The automorphism ρ_e maps $\mathfrak{G}^{\{v\}}$ to $\mathfrak{G}^{\{w\}}$, which is highlighted on the right.

A preordered graph $G^{\prec} = (V, E, \prec)$ is a graph $G = (V, E)$ equipped with a preorder $\prec$ on the vertices. For ease of notation, we identify the graph G with any given ordered version. A set of $\prec$-incomparable elements is called a *colour class*. We write C_i for the ith colour class in the linear order $\prec$ induces on the colour classes. A graph has colour classes of logarithmic size if $|C_i| \leq \log |V|$ for all colour classes C_i.

In the following, we present a definition and some properties of the Cai-Fürer-Immerman graphs. The CFI construction determines for each connected graph and each subset T of the vertex set a CFI-graph $\mathfrak{G}^T$. All $\mathfrak{G}^T$ fall into two isomorphism classes (*even* and *odd*). The CFI query then asks, given a CFI-graph, whether it is even or odd.

Given $G = (V, E)$, the CFI-graphs are certain subgraphs of the graph $\mathfrak{G}$ defined in the following. The graph $\mathfrak{G}$ contains, for each vertex $v \in V$, the *vertex gadget* $v^* = \{v^X : X \subseteq E(v)\}$ and, for each edge $e \in E$, the *edge gadget* $e^* = \{e^0, e^1\}$. The vertices v^X occurring in the vertex gadgets are called *inner vertices*. Let $\hat{V} = \cup_{v \in V} v^*$, $\hat{E} = \cup_{e \in E} e^*$, and for a subset $F \subseteq E$, $\hat{F} = \cup_{e \in F} e^*$. The edge set of $\mathfrak{G}$ is $\{\{v^X, e^1\} : v^X \in \hat{V}, e \in X\} \cup \{\{v^X, e^0\} : v^X \in \hat{V}, e \in E(v) \setminus X\}$.

Now let $T \subseteq V$. Then $v^*_T = \{v^X : |X| \text{ is odd}\}$ if $v \in T$ (then v^*_T is an odd gadget), $v^*_T = \{v^X : |X| \text{ is even}\}$ if $v \notin T$ (then v^*_T is even), and $\hat{V}_T = \cup_{v \in V} v^*_T$. The graph $\mathfrak{G}^T$ is the subgraph of $\mathfrak{G}$ induced by $\hat{E} \cup \bigcup_{v \in V} v^*_T$. We call $\mathfrak{G}^T$ even if $|T|$ is even, and odd otherwise. The construction is illustrated in Figure 1.

Consider an automorphism of $\mathfrak{G}$ that fixes v^* set-wise for each $v \in V$. Such an automorphism is always completely determined by a set $F \subseteq E$ such that for each $e \in F$, e^0 and e^1 are swapped. Formally, ρ_F is the mapping $e^i \mapsto e^{i-1}$ for $e \in F$ and $i \in \{0, 1\}$, $e^i \mapsto e^i$ for $e \notin F$, and $v^X \mapsto v^{X \triangle (F \cap E(v))}$. The group of automorphisms of $\mathfrak{G}$ that fix each v^* is generated by $\{\rho_e \coloneqq \rho_{\{e\}} : e \in E\}$.

As illustrated in Figure 1, each ρ_e maps $\mathfrak{G}^T$ to some $\mathfrak{G}^S$ such that T and S have the same parity. As shown by Cai, Fürer and Immerman [6], it follows that, for every connected graph G, $\mathfrak{G}^S \cong \mathfrak{G}^T$ if and only if $|S| \equiv |T| \pmod 2$. In other words, the even and odd CFI-graphs are uniquely determined up to isomorphism, and the automorphisms of $\mathfrak{G}$ stabilising all v^* are exactly the isomorphisms between graphs $\mathfrak{G}^T, \mathfrak{G}^S$ of the same parity.

The class of CFI-graphs over a graph class $\mathcal{C}$ is the class of all $\mathfrak{G}^T$ for $G \in \mathcal{C}$. If G is preordered by $\prec$, then $\prec$ induces an order on $\mathfrak{G}$ (and thus on all $\mathfrak{G}^T$) in the obvious way.

4 Computing the Parity of Cai-Fürer-Immerman Graphs over Ordered Graphs

A simple polynomial-time procedure for deciding the parity of a Cai-Fürer-Immerman graph just assigns to each vertex $e^i \in \hat{E}$ the label e^0 or e^1 and labels the vertices in $\hat{V}$ accordingly. By the nature of the automorphisms of $\mathfrak{G}$, this yields a graph $\mathfrak{G}^T$ that has the same parity as the input graph. Then T and thus the parity of $\mathfrak{G}^T$ can be determined easily.

A naive way to use that approach in CPT would require computing all the assignments of labels e^0 and e^1 in parallel and hence involve exponentially many sets, violating the polynomial resource bounds. However, the CPT procedure proposed by Dawar, Richerby and Rossman in [10] constructs an object which makes it possible to compute such an assignment in CPT using a linear order on the underlying graph.

Their construction represents, for each vertex v of the underlying graph, whether v is in T in the following way: τ_v contains the neighbourhoods of all inner vertices occurring in the given graph $\mathfrak{G}^T$, and $\tilde{\tau}_v$ contains the neighbourhoods of the remaining inner vertices of $\mathfrak{G}$. So each neighbourhood in τ_v contains an odd number of CFI vertices of the form e^1 if, and only if $v \in T$, if, and only if the number of e^1 in the neighbourhoods in $\tilde{\tau}_v$ is even.

Now consider an automorphism ρ of $\mathfrak{G}$ that maps $\mathfrak{G}^T$ to some $\mathfrak{G}^S$. Then, for each $v \in S \triangle T$, ρ swaps τ_v and $\tilde{\tau}_v$.

The aim is to compute the parity of $|T|$, which means to determine the number of τ_v whose elements contain an odd number of e^1 each, i.e. summing over all τ_v. Whenever an even number of τ_v is replaced by the respective $\tilde{\tau}_v$, the sum does not change, so, in particular, the sum is not affected by the automorphisms of $\mathfrak{G}$.

In order to compute the sum, the τ_v and $\tilde{\tau}_v$ are combined into sets $\mu_i, \tilde{\mu}_i$ for $i \leq |V|$ representing the parity of $|T|$ restricted to the first i vertices $v_1, \ldots, v_i$ (with respect to the linear order on V): $\mu_1 = \tau_{v_1}$, $\tilde{\mu}_1 = \tilde{\tau}_{v_1}$, $\mu_{i+1} = \{\langle \mu_i, \tau_{v_{i+1}} \rangle, \langle \tilde{\mu}_i, \tilde{\tau}_{v_{i+1}} \rangle\}$, and $\tilde{\mu}_{i+1} = \{\langle \mu_i, \tilde{\tau}_{v_{i+1}} \rangle, \langle \tilde{\mu}_i, \tau_{v_{i+1}} \rangle\}$. The algorithm computes the object $\mu_{|V|}$, which encodes the parity of $|T|$.

The $\mu_i, \tilde{\mu}_i$ can be viewed as representing sequences of $\tau_v, \tilde{\tau}_v$ such that the number of occurring $\tilde{\tau}_v$ is even in μ_i and odd in $\tilde{\mu}_i$. An automorphism of $\mathfrak{G}$ mapping $\mathfrak{G}^T$ to $\mathfrak{G}^S$ then changes an even number of τ_v to $\tilde{\tau}_v$ (since $S \triangle T$ is even) and thus can be shown to preserve $\mu_{|V|}$ and $\tilde{\mu}_{|V|}$. This property is called super-symmetry.

Super-symmetry means that whenever $e^0, e^1 \in e^*$ are swapped throughout the transitive closure of $\mu_{|V|}$, $\mu_{|V|}$ is fixed. Thus, assigning labels from $\{0, 1\}$ to a fixed edge gadget e^* in both possible ways in $\mu_{|V|}$ yields a unique result. These assignments can be computed sequentially using the linear order, resulting in a modified set where all neighbourhoods in τ_v contain an odd number of 1s if, and only if $v \in T$. From this object, the parity of $|T|$ can be computed.

5 Graphs with Colour Classes of Logarithmic Size

The algorithm from [10] processes the CFI-graph according to the linear order on the underlying graph. We generalise that construction by defining objects that correspond to multiple linear orders consistent with a given preorder, and obtain

▶ **Theorem 1.** *Let $\mathcal{K}$ be the class of connected, preordered graphs $G = (V, E, \prec)$ such that the size of each colour class is bounded by $\log |V|$. The CFI query over $\mathcal{K}$ is definable in Choiceless Polynomial Time.*

Without the linear order, there is no unique notion of the sets μ_i, $\tilde{\mu}_i$ described above that represent the parity of $|T \cap \{v_1, \dots, v_i\}|$. Instead, we define sets $\mu_M, \tilde{\mu}_M$ encoding the parity of $|T \cap M|$ for more general subsets $M \subseteq V$. The number of considered subsets M is kept small using the preorder. First, we construct objects for all subsets of the first colour class C_1, and obtain sets μ_{C_1} and $\tilde{\mu}_{C_1}$ for the full colour class. These are combined with all subsets of the second colour class, which yields $\mu_{C_1 \cup C_2}$, and so on for the remaining colour classes.

But the encoding of the μ_M, $\tilde{\mu}_M$ should again represent adding the value of a specific τ_v to a previous value. Therefore, for each subset $M \subseteq V$ for which μ_M and $\tilde{\mu}_M$ are constructed, all ways to decompose M as $M = N \uplus \{v\}$ for some v in the current colour class are considered.

To replace the e^i with their labels from $\{0, 1\}$, the algorithm in [10] also uses the linear order on the vertices. Because this is not possible with only a preorder, we already assign labels to the edge gadgets during the construction of the μ_M, $\tilde{\mu}_M$. More precisely, when viewing M as $N \uplus \{v\}$, all edges in the cut $(N, \{v\})$ are processed. So our CPT algorithm does not actually compute the μ_M and $\tilde{\mu}_M$, but modified objects ν_M, $\tilde{\nu}_M$ where all e^i for $e \in E[M]$ in the subgraph induced by M are already processed. Finally, the parity of T is extracted from the object ν_V using a parity function p.

Next, we describe these steps in detail. The sets τ_v, $\tilde{\tau}_v$ for $v \in V$ are defined as in [10], with one difference: To allow for assigning labels 0 or 1 to several vertices at once later without losing information, we equip each neighbourhood with a parity check bit, which will encode the parity of the number of edges that are labelled 1.

▶ **Definition 2.** Let $v \in V$. Then $\tau_v^T = \{\langle N(v^X), 0\rangle : v^X \in v_T^*\}$ and $\tilde{\tau}_v^T = \{\langle N(v^X), 0\rangle : v^X \in v^* \setminus v_T^*\}$, where $N(v^X)$ is the neighbourhood of v^X in the full graph $\mathfrak{G}$.

We omit the superscript T whenever it is clear from the context.

The τ_v, $\tilde{\tau}_v$ are combined to obtain objects μ_M, $\tilde{\mu}_M$ for certain subsets $M \subseteq V$ consistent with the preorder.

▶ **Definition 3.** Let $\mathcal{M}_i$ be the set of all $M \subseteq V$ such that $\bigcup_{j<i} C_j \subseteq M$ and $M \subseteq \bigcup_{j \leq i} C_j$. Then let $\mathcal{M}$ be the set of all $M \subseteq V$ that are in $\mathcal{M}_i$ for some i.

For subsets $M \in \mathcal{M}$, the objects $\mu_M, \tilde{\mu}_M$ are constructed inductively as follows.

▶ **Definition 4.** Let $T \subseteq V$, $v \in V$, $M \in \mathcal{M}$ and $v \notin M$. Then $\mu_{\{v\}}^T = \tau_v^T$, $\tilde{\mu}_{\{v\}}^T = \tilde{\tau}_v^T$, $\mu_{M,v}^T = \{\langle \mu_M^T, \tau_v^T\rangle, \langle \tilde{\mu}_M^T, \tilde{\tau}_v^T\rangle\}$, $\tilde{\mu}_{M,v}^T = \{\langle \mu_M^T, \tilde{\tau}_v^T\rangle, \langle \tilde{\mu}_M^T, \tau_v^T\rangle\}$, and, for $|M| > 1$, $\mu_M^T = \{\mu_{N,w}^T : N \in \mathcal{M} \text{ and } N \uplus \{w\} = M\}$ and $\tilde{\mu}_M^T = \{\tilde{\mu}_{N,w}^T : N \in \mathcal{M} \text{ and } N \uplus \{w\} = M\}$.

Like the μ_i^T, $\tilde{\mu}_i^T$ in the procedure by Dawar, Richerby and Rossman, the μ_M^T, $\tilde{\mu}_M^T$ characterise the parity of $|T \cap M|$, which can be shown by an easy induction on $|M|$.

▶ **Lemma 5.** $\mu_M^T = \mu_M^S \Leftrightarrow \tilde{\mu}_M^T = \tilde{\mu}_M^S \Leftrightarrow |S \cap M| \equiv |T \cap M| \pmod 2$ *and*
$\mu_M^T = \tilde{\mu}_M^S \Leftrightarrow \tilde{\mu}_M^T = \mu_M^S \Leftrightarrow |S \cap M| \not\equiv |T \cap M| \pmod 2$.

Next we define the objects ν_M, $\tilde{\nu}_M$ obtained by labelling the edge gadgets corresponding to the edges in the cut $(N, \{v\})$ when combining an object for a smaller set N with τ_v or $\tilde{\tau}_v$. For fixed v, a sufficiently small set of such mappings is determined by the vertices in v^*: Each v^X defines through its neighbourhood a unique vertex from each adjacent edge gadget. This allows to define, for each pair v^X, M, a mapping labelling edge vertices with binary values and aggregating these values in the parity check bit.

▶ **Definition 6.** Let $v^X \in \hat{V}$ and $M \subseteq V$. Then, for each set z and $i \in \{0, 1\}$, let $B_{v^X,M}(\langle z, i\rangle) = \langle z \setminus (\widehat{M, \{v\}}), j\rangle$, where $j = i + |(\widehat{M, \{v\}}) \cap N(v^X)| \mod 2$, and, for any other set y, $B_{v^X,M}(y) = \{B_{v^X,M}(z) : z \in y\}$.

Recall that $\widehat{(M, \{v\})}$ denotes the set of vertices in edge gadgets corresponding to edges in the cut $(M, \{v\})$.

Like the mappings from [10] that label the e^i by i or $1-i$ for a single edge $e \in E$, the $B_{v^X,M}$ are computed simultaneously for all $v^X \in v^*$. By similar symmetry arguments, this yields a unique object.

▶ **Lemma 7.** *If $z \in \mathrm{HF}(\hat{E})$ is fixed by the automorphisms ρ_e for all $e \in (M, \{v\})$, then $B_{v^X,M}(z) = B_{v^Y,M}(z)$ for any $X, Y \subseteq E(v)$ and $B_{v^X,M}(z)$ is fixed by the same automorphisms.*

Proof. Let $X' = X \cap (M, \{v\})$ and $Y' = Y \cap (M, \{v\})$. Since $X' \triangle Y' \subseteq (M, \{v\})$, $z = \rho_{X' \triangle Y'}(z)$. By definition of $B_{v^X,M}$ and $B_{v^Y,M}$, $B_{v^X,M}(\rho_{X' \triangle Y'}(z)) = B_{v^Y,M}(z)$. Thus $B_{v^X,M}(z) = B_{v^X,M}(\rho_{X' \triangle Y'}(z)) = B_{v^Y,M}(z)$. Obviously ρ_e fixes $B_{v^X,M}(z)$ for $e \in (M, \{v\})$, because the corresponding CFI vertices do not occur in $B_{v^X,M}(z)$. ◀

The previous lemma justifies writing $B_{v,M}(z)$ for the mapping removing edges along the cut (M, v) if z is fixed by suitable automorphisms. Using these mappings, the objects ν_M, $\tilde{\nu}_M$ can be defined. The sets ν, $\tilde{\nu}$ for sets $\{v\}$ or pairs M, v are defined like the corresponding sets $\mu, \tilde{\mu}$, where μ is now replaced by ν. The only change occurs when aggregating ν_M and $\tilde{\nu}_M$: Then the corresponding mapping is applied to the $\nu_{N,v}$ and $\tilde{\nu}_{N,v}$

▶ **Definition 8.** Let $M \in \mathcal{M}$ with $|M| > 1$. $\nu_M = \{B_{v,N}(\nu_{N,v}) : N \uplus \{v\} = M \text{ and } N \in \mathcal{M}\}$ and $\tilde{\nu}_M = \{B_{v,N}(\tilde{\nu}_{N,v}) : N \uplus \{v\} = M \text{ and } N \in \mathcal{M}\}$.

To show that the $\nu_M, \tilde{\nu}_M$ are well-defined, we have to show that the construction only uses the mapping $B_{v,M}$ for sets that are fixed by all necessary automorphisms.

The proof uses that, like $\mu^T_M, \tilde{\mu}^T_M$, the sets $\nu^T_M, \tilde{\nu}^T_M$ characterise the parity of $|T \cap M|$. To show this, and make the $\nu^T_M, \tilde{\nu}^T_M$ more accessible to further analysis, we show that ν_M (resp. $\tilde{\nu}_M$) arises from μ_M (resp. $\tilde{\mu}_M$) by labelling and removing all edges in $E[M]$. That notion is formalised via mappings labelling arbitrary (sets of) edge gadgets in a way that is not CPT-definable in general, but makes it possible to reason about a specific replacement. We then show that, on the objects $\nu_M, \tilde{\nu}_M$, both kinds of mappings have the same effect.

▶ **Definition 9.** Let $e \in E$. Then $B_e(\langle z, i\rangle) = \langle z \setminus \{e\}, i + j\rangle$, where $j = 1$ if and only if $e^1 \in z$ and $+$ is addition modulo 2. For any other set y, $B_e(y) = \{B_e(z) : z \in y\}$. Let $F \subseteq E$ and let $e_1, \ldots, e_k$ be an enumeration of F. Then $B_F = B_{e_1} \circ \cdots \circ B_{e_k}$. Note that, for any $e \neq e'$, $B_e \circ B_{e'} = B_{e'} \circ B_e$, so B_F is well-defined.

Now the sets $\nu_M, \tilde{\nu}_M$ can be characterised as follows.

▶ **Lemma 10.** *Let $S, T \subseteq V$, $v \in V$ and $M, N \in \mathcal{M}$ such that $N \uplus \{v\} = M$.*

1. **(a)** $\nu^T_{N,v} = \nu^S_{N,v} \Leftrightarrow \tilde{\nu}^T_{N,v} = \tilde{\nu}^S_{N,v} \Leftrightarrow |T \cap M| \equiv |S \cap M| \pmod 2$,
   (b) $\tilde{\nu}^T_{N,v} = \nu^S_{N,v} \Leftrightarrow \nu^T_{N,v} = \tilde{\nu}^S_{N,v} \Leftrightarrow |T \cap M| \not\equiv |S \cap M| \pmod 2$.
2. *$\nu_{N,v}$, $\tilde{\nu}_{N,v}$ are fixed by the automorphism ρ_e for every $e \in (N, \{v\})$.*
3. *$\nu_M = B_{E[M]}(\tilde{\mu}_M)$ and $\tilde{\nu}_M = B_{E[M]}(\mu_M)$.*
4. **(a)** $\nu^T_M = \nu^S_M \Leftrightarrow \tilde{\nu}^T_M = \tilde{\nu}^S_M \Leftrightarrow |T \cap M| \equiv |S \cap M| \pmod 2$,
   (b) $\nu^T_M = \tilde{\nu}^S_M \Leftrightarrow \tilde{\nu}^T_M = \nu^S_M \Leftrightarrow |T \cap M| \not\equiv |S \cap M| \pmod 2$.

Proof. Simultaneously by induction on $|M|$. ◀

▶ **Corollary 11.** *ν_M is well-defined for all $M \in \mathcal{M}$.*

So the final set ν_V is a hereditarily finite set over $\{0, 1\}$ representing the parity of $|T|$. To extract that parity, we use the following aggregation function on $\mathrm{tc}(\nu_V)$:

▶ **Definition 12.**

$$p(x) = \begin{cases} i, & x = \langle N(v^X), i\rangle, \\ p(x_1) + p(x_2) \mod 2, & x = \langle x_1, x_2\rangle \text{ and } x_2 \notin \{0,1\}, \\ \prod_{y\in x} p(y), & x \text{ is a set.} \end{cases}$$

▶ **Lemma 13.** *$p(\nu_V^T) = 0$ if and only if $|T|$ is even.*

Proof. As, by Lemma 10, $\nu_V^T = B_E(\mu_V^T)$ and B_E labels the edges throughout the transitive closure, we can reason inductively about objects for smaller sets where *all* edges have been removed. Note that it suffices to consider the cases $T = \emptyset$ and $T = \{x\}$ for some $x \in V$. Thus the lemma is shown if the following statements are verified for all $M \subseteq V$ and $v \in V \setminus M$, where $P(x)$ denotes $p(B_E(x))$:

- $P(\mu_M^\emptyset) = P(\mu_{M,v}^\emptyset) = 0$, $P(\tilde{\mu}_M^\emptyset) = P(\tilde{\mu}_{M,v}^\emptyset) = 1$,
- $P(\mu_{M,v}^{\{x\}}) = 1 \Leftrightarrow P(\tilde{\mu}_{M,v}^{\{x\}}) = 0 \Leftrightarrow x \in M \cup \{v\}$,
- $P(\mu_M^{\{x\}}) = 1 \Leftrightarrow P(\tilde{\mu}_M^{\{x\}}) = 0 \Leftrightarrow x \in M$.

The statements can be shown by induction on $|M|$, and follow from the definition of the objects $\tau_v^T, \tilde{\tau}_v^T, \mu_M^T, \tilde{\mu}_M^T$ and the mappings B_E and p. ◀

It remains to show that the construction is CPT-definable. All sets used in the computation can be defined by the set-theoretic operations available in CPT. Thus we only need to prove that the construction does not use too many or too complex objects.

▶ **Lemma 14.** *$|\operatorname{tc}(\mu_M^T)|$ is polynomial in $|\mathfrak{G}^T|$ for $M, T \subseteq V$.*

Proof. First note that the sets $\mu_M, \tilde{\mu}_M$ for $|M| = k+1$ are constructed from objects created for sets of size k, so it suffices to count the newly created sets for $|M| = k+1$. The sets $\mu_{M,v}$, $\tilde{\mu}_{M,v}$ are constructed for at most 2^{C_i} many sets and $|C_i|$ many vertices for some colour class C_i of logarithmic size. Furthermore, there are also at most $2^{|C_i|}$ many new sets μ_M and $\tilde{\mu}_M$, which are built from $\mu_{M,v}$ and $\tilde{\mu}_{M,v}$. ◀

By Lemma 10, $|\operatorname{tc}(\nu_M)| \leq |\operatorname{tc}(\mu_M)|$ for all $M \subseteq V$. So our construction is CPT-definable, which completes the proof of Theorem 1.

6 Classes of Unordered Graphs with Linear Maximal Degree

The techniques used for graphs with colour classes of logarithmic size can be further refined for classes of graphs that do not possess any order, but where the maximal degree is linear in the size of the graph. Note that this implies that the CFI-query over, for example, complete graphs is CPT-definable, verifying a claim from [10].

▶ **Theorem 15.** *For every $k \in \mathbb{N}$, the CFI query over the graphs $G = (V, E)$ with maximal degree $\geq \frac{|V|}{k}$ is definable in* CPT *using only sets of constant rank not depending on k.*

We now explain the modifications made to adapt the procedure presented in the previous section to these graph classes.

As previously, we start with an object τ_v for each $v \in V$, that encodes whether $v \in T$. In the case of logarithmic colour classes, the number of subsets for which the "sum" of the τ_v is computed is kept small by using the preorder. The set of these subsets can be defined in CPT. For unordered graphs, CPT cannot distinguish between any two equally sized subsets of the vertex set. However, the CFI-graphs graphs are large enough to permit considering *all*

subsets of V, because for the vertex v with maximal degree, each v^X corresponds to a subset of v's $\geq \frac{|V|}{k}$ many neighbours.

Recall that, in the algorithm in [10], sequences of τ_v and $\tilde{\tau}_v$ with an even number of $\tilde{\tau}_v$ are encoded. With access to all subsets of V, these sequences can be replaced by all sets of τ_v and $\tilde{\tau}_v$ containing an even number of $\tilde{\tau}_v$. The sets μ_M, $\tilde{\mu}_M$ for $M \subseteq V$ defined in that way again characterise the parity of $|T \cap M|$.

We again use the mappings $B_{v,M}$ arising from Lemma 7 to remove edges along cuts (M, v). However, these mappings require a vertex v to be fixed during the construction. So we formalise the construction of the μ_M, $\tilde{\mu}_M$ as successively enlarging the sets M by a single vertex. For instance, whenever $M = N \uplus \{v\}$, each set in μ_M containing an even number of $\tilde{\tau}_w$ can be constructed by adding τ_v to a set in μ_N, or $\tilde{\tau}_v$ to a set in $\tilde{\mu}_M$. Since the choice of N and v does not affect the resulting sets, μ_M can be computed by simultaneously using all these decompositions of M.

So the sets ν_M with labelled edges are constructed by splitting M into all possible decompositions $N \uplus \{v\}$, adding τ_v (resp. $\tilde{\tau}_V$) to the sets in μ_N ($\tilde{\mu}_N$) and processing the edges along the cut $(N, \{v\})$.

Analogously to Lemma 10, we can show that the sets $\nu_M, \tilde{\nu}_M$ are well-defined, again using the mappings B_F from Definition 9, that remove edges in a specific set $F \subseteq E$. It also follows that ν_M is well-defined for all $M \subseteq V$.

The parity of $|T|$ is computed with a new aggregation function. It also first extracts and multiplies the parity bits from the neighbourhoods in each τ_v. Since the parity of the neighbourhoods in τ_v, $\tilde{\tau}_v$ determines whether v^* is an odd gadget, the product is 1 for τ_v if and only if v^* is odd. Next, the sum of the parities (modulo 2) of each set in ν_V consisting of an even number of $\tilde{\tau}_v$ is computed. By definition, that sum is even for every such set if and only if the number of odd vertex gadgets is even. So it is possible to extract the parity of $|T|$ from ν_V.

It remains to show that the ν_M can be constructed in Choiceless Polynomial Time. One of the main obstacles is to compute sums modulo 2 when applying the mappings $B_{v,M}$ and p, because counting would in general require ordinals, which do not have constant rank. Furthermore, we need to show that the size of the transitive closure of each ν_M is polynomial in $|\mathfrak{G}|$.

▶ **Lemma 16.** *For any definable set $x \in \mathrm{HF}(\mathfrak{A})$ of rank k with $|x|$ logarithmic in $|\mathfrak{A}|$, the parity of $|x|$ can be computed in* CPT *over $\mathfrak{A}$ with sets of rank k.*

First note that the size of each neighbourhood occurring in the transitive closure of each ν_M is bounded by $|V|$ and thus logarithmic in the size of the CFI-graph, so the lemma can be applied to the computation of the mappings $B_{v,M}$.

Proof. For all $n \leq |x|$, compute the set of all subsets of x of size n and store their parity. Start with the set of all singletons and parity 1. Given sets of size n, construct the sets of size $n+1$ analogously to the construction of the μ_M and flip the parity. As soon as the set $\{x\}$ itself has been constructed, the parity of $|x|$ can be extracted. ◀

▶ **Lemma 17.** *If G has a vertex of degree $\geq \frac{|V|}{k}$, ν_V^T can be constructed in Choiceless Polynomial Time from the input $\mathfrak{G}^T$.*

Proof. Each set ν_M or $\tilde{\nu}_M$ constructed by the algorithm contains $\leq 2^{|M|}$ many sets consisting of (polynomially sized) sets τ_v and $\tilde{\tau}_v$, and there are $2^{|V|}$ subsets M. The number and size of the ν_M and $\tilde{\nu}_M$ is polynomial in $|\mathfrak{G}^T|$ because some $v \in V$ has $\geq \frac{|V|}{k}$ neighbours, so the

vertex gadget v_T^* is of size $2^{\frac{|V|}{k}-1}$. So the transitive closure of all constructed objects is of polynomial size. ◀

7 Computations Are More Powerful Over Sets Than Over Tuples

In this section we want to show that every CPT-program which decides the Cai-Fürer-Immerman query over complete graphs (where the maximal degree is obviously linear) has to use set-like objects.

Of course, the immediate question is: what are set-like objects?

In order to motivate our following definition, let us consider a simple example. Let $G = (V, E)$ be a complete graph over a set V with $|V| = n$ vertices. The automorphism group of G is the full symmetric group $\Gamma = \mathrm{Sym}(V)$. Moreover, let $x = \{a_0, \dots, a_{k-1}\} \subseteq V$ be a *set* consisting of k distinct vertices, and let $y = (a_0, \dots, a_{k-1}) \in V^k$ be some arrangement of these k vertices as a *k-tuple* over V. Since every CPT-computation on G is invariant under the action of Γ, every CPT-program which constructs x also has to completely construct $\mathrm{Orbit}(x) = \{\pi(x) : \pi \in \Gamma\}$ of x (the same holds for every program which constructs y).

The question is, what do we know about the sizes of the orbits of x and y? It is easy to see that $|\mathrm{Orbit}(x)| = \binom{n}{k}$ while $|\mathrm{Orbit}(y)| = \binom{n}{k} \cdot k!$. Hence, for large k the orbit of the *set* x is much smaller than the size of the orbit of the *tuple* y (although they are defined over the same set of atoms). To put it differently, for large k the size of the stabiliser $\mathrm{Stab}(x)$ of x is much larger than the size of the stabiliser $\mathrm{Stab}(y)$ of y. Indeed, $|\mathrm{Stab}(x)| = k! \cdot (n-k)!$ and $|\mathrm{Stab}(y)| = (n-k)!$. Obviously, the reason is that in order to fix the tuple y we have to fix *every* entry a_i *point-wise* whereas we can permute the elements $\{a_0, \dots, a_{k-1}\}$ while keeping x fixed. In algebraic terms, the point-wise stabiliser $\mathrm{Stab}^\bullet\{a_1, \dots, a_k\} = \{\pi : \pi(a_i) = a_i \text{ for all } i < k\}$ of $a_0, \dots, a_{k-1}$ coincides with $\mathrm{Stab}(y)$ while it is a subgroup of $\mathrm{Stab}(x)$ of index $k!$.

▶ **Definition 18.** Let $\mathfrak{A}$ be a structure with automorphism group $\Gamma = \mathrm{Aut}(\mathfrak{A})$. Let $x \in \mathrm{HF}(A)$. A set $\sigma \subseteq A$ of atoms is a *support* for x if $\mathrm{Stab}^\bullet(\sigma) \leq \mathrm{Stab}(x)$, and σ is called a *strong support* of x if $\mathrm{Stab}^\bullet(\sigma) = \mathrm{Stab}(x)$.

Accordingly, we say that an element $x \in \mathrm{HF}(A)$ is *strongly supported* if it has a strong support. For example, every atom $a \in A$ is strongly supported and every $x \in \mathrm{HF}(A)$ with $\mathrm{Stab}(x) = \Gamma$ has the strong support $\emptyset$. Note that for the above example, the set $\{a_0, \dots, a_{k-1}\}$ is a support for the *set* $x = \{a_0, \dots, a_{k-1}\}$ and it is a *strong support* for the *tuple* $y = (a_0, \dots, a_{k-1})$. The notion of strongly supported sets is taken as a working definition for objects which are "not set-like", or more intuitively, which are "sequence-like". Those objects enforce an inherent order on the supporting atoms. We aim to prove Theorem 23: no CPT-program which can only access strongly supported objects can express the CFI query over complete graphs.

As a starting point for the analysis of strong supports, we examine the structure of the automorphism group of $\mathfrak{G}^T$.

▶ **Lemma 19** (Action of $\mathrm{Sym}(V)$)**.** *Let $G = (V, E)$ be a complete graph and let $S_V = \mathrm{Aut}(G) = \mathrm{Sym}(V)$. Then every $\pi \in S_V$ can be lifted to an automorphism $\rho \in \Gamma = \mathrm{Aut}(\mathfrak{G}^T)$ such that $\rho(v^*) = \rho(w^*)$ if and only if $\pi(v) = \pi(w)$ for all $v, w \in V$.*

Proof. Constructing ρ for a transposition $(v, w) \in S_V$, one has to make sure that all edge gadgets $\{u, v\}$ and $\{u, w\}$, and their neighbours in the vertex gadgets, are exchanged, and the parity of v^* and w^* is exchanged if necessary. ◀

The automorphism group $\Gamma = \mathrm{Aut}(\mathfrak{G}^T)$ of a CFI-graph over a complete graph $G = (V, E)$ can be decomposed as follows. Let us denote by Δ the subgroup of Γ which stabilises all sets of inner vertices v^* for $v \in V$, i.e. $\Delta = \bigcap_{v \in V} \mathrm{Stab}(v^*)$. Then Δ is a normal subgroup of Γ which can be identified with a subgroup of $\mathbb{Z}_2^{|E|}$ and which coincides with the CFI-automorphism group that one obtains if the underlying complete graph is ordered. It then follows, by the preceding lemma, that $\Gamma/\Delta \cong S_V$. We will often identify S_V with this group and also the corresponding actions of Γ/Δ on $\{v^* : v \in V\}$ and of S_V on V.

Next, we explain the idea of our proof, which is based on the techniques developed by Dawar, Richerby and Rossman in [10]. It is well-known that every CPT-program Π can be translated into a formula φ_Π of infinitary logic with counting and k variables, for a certain k which depends on Π (we denote this logic by $C^k_{\infty\omega}$), with the following property: for every input structure $\mathfrak{A}$, the formula φ_Π is equivalent to Π in the sense that

$$\mathfrak{A} \models \Pi \Leftrightarrow \mathrm{Active}(\Pi, \mathfrak{A}) \models \varphi_\Pi,$$

where $\mathrm{Active}(\Pi, \mathfrak{A})$ is the extension of $\mathfrak{A}$ by all hereditarily finite sets which are *activated* (intuitively: have to be constructed) during the run of Π on $\mathfrak{A}$. For details see [10, 3, 4]. Hence, if we can show for two structures $\mathfrak{A}$ and $\mathfrak{B}$ that $\mathrm{Active}(\Pi, \mathfrak{A})$ and $\mathrm{Active}(\Pi, \mathfrak{B})$ cannot be distinguished by any $C^k_{\infty\omega}$-formula ($\mathrm{Active}(\Pi, \mathfrak{A}) \equiv^C_k \mathrm{Active}(\Pi, \mathfrak{B})$), then we can conclude that Π cannot distinguish between $\mathfrak{A}$ and $\mathfrak{B}$ as well. However, there are two serious difficulties in this approach:

- Showing that $\mathrm{Active}(\Pi, \mathfrak{B}) \equiv^C_k \mathrm{Active}(\Pi, \mathfrak{B})$ is combinatorially extremely challenging, because we have to relate highly-nested sets over the two input structures.
- The structures $\mathrm{Active}(\Pi, \mathfrak{A})$ and $\mathrm{Active}(\Pi, \mathfrak{B})$ depend on the program Π. This is problematic as it would require to show $C^k_{\infty\omega}$-equivalence of structures which are defined rather indirectly by the run of the CPT-program Π.

There is a very nice approach, which was first used by Blass, Gurevich and Shelah in [3] and later refined and generalised by Dawar, Richerby and Rossman in [10, 18] which solves both problems at once. Very roughly, the idea is that for certain structures $\mathfrak{A}$ we can over-approximate $\mathrm{Active}(\Pi, \mathfrak{A})$ by the structure $\mathrm{HF}(\mathfrak{A})_\ell$ consisting of all hereditarily finite sets which are *transitively ℓ-supported*, that is those sets which have a support of size at most ℓ and whose transitive closure only consists of sets which again have a support of size at most ℓ. Specifically, if we can prove that *all* sets which can be activated in a run of Π on $\mathfrak{A}$ are ℓ-supported, then we know that $\mathrm{Active}(\Pi, \mathfrak{A}) \subseteq \mathrm{HF}(\mathfrak{A})_\ell$. More importantly, if the orbits of tuples (of sufficient length) in the structure $\mathfrak{A}$ are definable in $C^m_{\infty\omega}$ for a certain m ($\mathfrak{A}$ is then called C^m-homogeneous), then we can obtain a description of the possibly highly-nested transitively ℓ-supported sets in terms of their supports (which are lists of atoms) and syntactic objects which do not depend on $\mathfrak{A}$. This reduces the complexity to deal with highly nested sets in $\mathrm{HF}(\mathfrak{A})$ to flat objects over $\mathfrak{A}$. Our aim is to adapt the approach of Dawar, Richerby and Rossman (Theorem 22) for our application. To this end CFI-graphs over complete graphs should satisfy the following requirements:

- they are C^m-homogeneous for a certain m (and large enough tuples), this is Lemma 20, and
- all objects which can be activated by a CPT-program that only uses strongly supported sets have small supports (Lemma 21).

▶ **Lemma 20** (C^m-homogeneity). *Let $n, m \in \mathbb{N}$ such that $3 \leq m \leq n - 3$. Let $\mathfrak{G}^T$ be a CFI-graph over a complete graph $G = (V, E)$ with $|V| = n$. Let $\vec{a}$, $\vec{b}$ be tuples of vertices of $\mathfrak{G}^T$ both of length i for some $i \leq m - 3$. Then, $(\mathfrak{G}^T, \vec{a}) \equiv^C_m (\mathfrak{G}^T, \vec{b})$ implies $Orbit(\vec{a}) = Orbit(\vec{b})$.*

*In other words, CFI-graphs over complete graphs with n vertices are C^m-*homogeneous *(with respect to tuples of length $i \leq m-3$).*

▶ **Lemma 21** (Sizes of supports). *For all $q \geq 1$, $0 < \epsilon < 1$, we can find $m \geq 1$ such that for all strongly supported sets $x \in \mathrm{HF}(\hat{V}_T \cup \hat{E})$ over CFI-graphs $\mathfrak{G}^T$ of complete graphs $G = (V, E)$ with $|V| = n \geq m$ vertices we have*

- *$|Orbit(x)| > 2^{n \cdot q}$ or*
- *$|\sigma| \leq \epsilon \cdot n$ for some (not necessarily strong) support σ for x.*

In other words, all strongly supported sets $x \in \mathrm{HF}(\hat{V}_T \cup \hat{E})$ with small orbit $|\mathrm{Orbit}(x)| \leq 2^{n \cdot q}$ are $(\epsilon \cdot n)$-supported.

Proof. We assume that $|\mathrm{Orbit}(x)| \leq 2^{n \cdot q}$ and choose a strong support $\sigma \subseteq \hat{V}_T \cup \hat{E}$ for x which is minimal with respect to $\subseteq$. From the orbit stabiliser theorem, the fact that $S_V \leq \Gamma$ (Lemma 19), and since σ is a strong support for x it follows that

$$\frac{|S_V|}{|\mathrm{Stab}^{\bullet}_{S_V}(\sigma)|} \leq \frac{|\Gamma|}{|\mathrm{Stab}^{\bullet}_{\Gamma}(\sigma)|} = \frac{|\Gamma|}{|\mathrm{Stab}_{\Gamma}(x)|} = |\mathrm{Orbit}(x)| \leq 2^{n \cdot q}. \tag{1}$$

Our next aim is to show that the support σ induces a partition of V as $V = V_0 \uplus V_1 \uplus \cdots \uplus V_\ell$, such that every automorphism $\pi \in \mathrm{Stab}^{\bullet}_{S_V}(\sigma)$ respects this partition. More precisely, the following holds:

- $\mathrm{Stab}^{\bullet}_{S_V}(\sigma) \leq \mathrm{Stab}_{S_V}(\bigcup_{v \in V_0} v^*_T) \cap \cdots \cap \mathrm{Stab}_{S_V}(\bigcup_{v \in V_\ell} v^*_T)$, and
- $\ell = |\sigma \cap \hat{V}_T|$, and
- using the elements in σ as parameters one can define each of the sets V_i (inside the structure $\mathfrak{G}^T$) using a $C^{\ell+2}_{\infty\omega}$-formula.

To simplify our notation, we set $V^*_i = \bigcup_{v \in V_i} v^*_T$. To start, we first define $W \subseteq V$ as the set of vertices $v \in V$ such that σ contains an inner node from the CFI-gadget associated with the vertex v, that is $\sigma \cap v^*_T \neq \emptyset$. Then, clearly, $\mathrm{Stab}^{\bullet}_{S_V}(\sigma) \leq \bigcap_{w \in W} \mathrm{Stab}_{S_V}(\{w^*_T\}) \cap \mathrm{Stab}_{S_V}((V \setminus W)^*)$ and we obtain a first partition of V into $|W|+1$ many blocks as $V = \biguplus_{w \in W}\{w\} \uplus V \setminus W$.

For all $w \in W$, we choose a witnessing element $a_w \in \sigma \cap w^*_T$. We next want to show that whenever we fix an additional element $b \in \sigma \cap w^*_T$ for some $w \in W$ and $b \neq a_w$, then we obtain a refined partition of V which still satisfies the above properties. We show this by induction on the number $i = 0, \ldots, |\sigma \cap \hat{V}^T| - |W|$ of processed inner nodes which are different from the a_w. Thus, let us assume that we have already considered the elements in $\sigma_i = \{a_w, b_1, \ldots, b_i : w \in W\} \subseteq \sigma \cap \hat{V}^T$ and obtained a refined partition of V as $V = \biguplus_{w \in W}\{w\} \uplus V_1 \uplus \cdots \uplus V_i \uplus V \setminus (\bigcup_{j=1}^{i} V_j \cup W)$. Choose $b \in (\sigma \cap w^*_T) \setminus \sigma_i$ for some $w \in W$. Then $a_w = w^X$ and $b = w^Y$ for some sets $X, Y \subseteq E(w)$. Then the set of edges in $X \triangle Y \subseteq E(w)$ is $C^{\ell+2}_{\infty\omega}$-definable and hence also the set of neighbours $Z \subseteq V$ of w which are incident with these edges. Assume that Z is a union of blocks from the old partition. Then b would already be definable from the other parameters, a contradiction to the minimality of σ. Hence, the partition can indeed be refined and the new blocks are again $C^{\ell+2}_{\infty\omega}$-definable.

Next we obtain a bound on $\ell = |\sigma \cap \hat{V}^T|$. We know from Equation 1 that

$$\left|S_V / \mathrm{Stab}^{\bullet}_{S_V}(\sigma)\right| \leq 2^{n \cdot q},$$

and also that $\mathrm{Stab}^{\bullet}_{S_V}(\sigma) \leq \mathrm{Stab}_{S_V}(V^*_0) \cap \cdots \cap \mathrm{Stab}_{S_V}(V^*_\ell)$. Moreover, it is easy to see that $\frac{n!}{(n-\ell)!} \leq |S_V / (\mathrm{Stab}_{S_V}(V^*_0) \cap \cdots \cap \mathrm{Stab}_{S_V}(V^*_\ell))|$. It follows that $n!/(n-\ell)! \leq 2^{n \cdot q}$.

By the Stirling formula we can find a constant $c > 0$ such that

$$\ell \leq \frac{q}{(\log n + \log e)} \cdot n - \frac{\log c}{(\log n + \log e)}.$$

Hence, for large enough n it follows that $\ell \leq \epsilon \cdot n$. More generally, the argument shows that the number ℓ of parts in a partition of V as above is, for large n, smaller than $\epsilon \cdot n$. However, to prove our original claim we have to find a support σ of x such that $|\sigma| \leq \epsilon \cdot n$ (by now we only proved that $\ell = |\sigma \cap \hat{V}^T| \leq \epsilon \cdot n$).

Hence, in a second step we consider the elements in σ which are part of $\hat{E}$. Every such element $e^i \in \sigma \cap \hat{E}$ corresponds to an edge $e = \{v, w\}$ in the original complete graph. As above, we can use the e^i to refine the partition of V, because we can define (using e^i) the new block $\{v, w\}$. However, in contrast to the case above, it might happen that the partition cannot be refined although we still have unprocessed $e^i \in \sigma \cap \hat{E}$ left. There are two possible reasons: either we already identified a block $\{v, w\}$ (for instance, by using inner nodes) or we have refined V into one block $\{v\}$ and another block $\{w\}$. We overcome this issue by modifying the strong support σ. In fact, we can replace each $e^i \in \sigma$ where $e = \{v, w\}$ by two inner nodes $a \in v_T^*$ and $b \in w_T^*$ to obtain a new support σ' for x, i.e. $\mathrm{Stab}^\bullet_\Gamma(\sigma') \leq \mathrm{Stab}^\bullet_\Gamma(\sigma) = \mathrm{Stab}(x)$. Note, however, that σ' might no longer be a strong support for x. The question remains how many inner nodes we have to add to cover all edges in the support for which we do not obtain a refinement of our partition. The answer is: not more than we have blocks in our partition after a maximal refinement. This is because the only cases where edges do not lead to refinements of the partition can be resolved by fixing an inner node for one vertex in a block of size one or two. Hence, we can find a new support σ' for x of size $|\sigma'| \leq 2 \cdot \ell$ where ℓ denotes the maximal length of a partition of V that we can get by using first all inner nodes and then some (maximal set of) edge nodes. Since we can choose n large enough such that $\ell \leq \frac{\epsilon}{2} \cdot n$, this suffices to prove our claim. ◀

▶ **Theorem 22** (Dawar, Richerby, Rossman). *Let $\ell \geq 1$, $k \geq 1$ and let $\mathfrak{A}$ and $\mathfrak{B}$ be two $C^{\ell \cdot k}$-homogeneous structures. If $\mathfrak{A} \equiv^C_{\ell \cdot k} \mathfrak{B}$, then $\mathrm{HF}(\mathfrak{A})_\ell \equiv^C_k \mathrm{HF}(\mathfrak{B})_\ell$.*

▶ **Theorem 23.** *Let $\Pi \in CPT$ be a program which activates only strongly supported sets and which has resource bounds $p(n) = n^q$ for some $q \geq 1$. Then Π cannot decide the Cai-Fürer-Immerman query over complete graphs.*

Proof. To obtain a contradiction, assume that there is a CPT-program Π with the above properties. First we translate Π into an equivalent $C^k_{\infty\omega}$-formula φ_Π which simulates Π on $\mathrm{Active}(\Pi, \mathfrak{G}^T)$, i.e. $\mathfrak{G}^T \models \Pi$ if, and only if, $\mathrm{Active}(\Pi, \mathfrak{G}^T) \models \varphi_\Pi$.

Secondly, we choose $m \geq 1$ sufficiently large (according to Lemma 21 for q and $\epsilon = \frac{1}{2k}$) such that all sets $x \in \mathrm{HF}(\mathfrak{G}^T)$ which can be activated by Π in CFI-graphs $\mathfrak{G}^T$ over complete graphs $G = (V, E)$ with $|V| = n \geq m$ vertices have a support of size at most $\frac{1}{2k} \cdot n = \epsilon \cdot n$. Hence $\mathrm{Active}(\Pi, \mathfrak{G}^T) \subseteq \mathrm{HF}(\mathfrak{G}^T)_{\epsilon \cdot n}$. Let $\mathfrak{G}^T$ be an even and $\mathfrak{G}^S$ be an odd CFI-graph over such a complete graph G. It is well-known [6] that $\mathfrak{G}^T \equiv^C_{n/2} \mathfrak{G}^S$. Since $n/2 = \epsilon \cdot n \cdot k$, we can apply Theorem 22 to conclude that $\mathrm{HF}(\mathfrak{G}^T)_{\epsilon \cdot n} \equiv^C_k \mathrm{HF}(\mathfrak{G}^S)_{\epsilon \cdot n}$. Hence, Π cannot distinguish between the two CFI-graphs. ◀

This result provides a deeper understanding of the expressive power of fragments of Choiceless Polynomial Time. Interpretation Logic, which was introduced in [12], characterises CPT and some of its fragments in terms of first-order interpretations (with the Härtig quantifier for equicardinality testing). This framework allows to iterate a first-order interpretation until a first-order halting condition is satisfied, and evaluates a first-order sentence on the resulting structure (for details, see [12]), with polynomial bounds on the number of iterations and the size of the intermediate structures. The fragment of interpretation logic that arises when only interpretations without congruences are allowed, i.e. different tuples cannot be identified to obtain a single element of the interpreted structure, has already been shown to be strictly

weaker than CPT. Without counting (respectively equicardinality quantifiers), this fragment corresponds to the database query language $\text{while}_{\text{new}}$, which permits construction of new elements for definable *tuples*. For the fragment with counting, it could be shown [12] that, on structures with bounded colour class size, only CPT computations with sets of bounded rank can be simulated. Theorem 23 now implies that the fragment not only fails to define sets of unbounded rank, but also any kind of set-like objects.

▶ **Corollary 24.** *Polynomial-Time Interpretation Logic without congruences cannot define the CFI query over complete graphs.*

Proof. A CPT-program simulating an Interpretation Logic computation constructs exactly those sets that represent the interpreted structures, i.e. a polynomially sized, first-order definable set of *tuples* for each structure, and relations in the structures. These objects are strongly supported. ◀

8 Conclusion and Future Work

We generalise previous expressibility results for the CFI query in Choiceless Polynomial Time to graphs with colour classes of logarithmic size and graph classes where the maximal degree is linear. In the latter case, CPT over sets of bounded rank suffices, which illustrates that on large enough input structures, the full power of CPT is not exhausted even for structures that, like the CFI-graphs, have many symmetries. The first result suggests that results for structures with bounded colour class size may be lifted to larger colour classes.

Clearly, a CPT procedure for the CFI query over arbitrary graphs remains desirable. Especially since graph classes with vertices of large (linear) degree are already covered, graphs with bounded degree, especially the possibly simpler case of 3-regular graphs, seem to be an important benchmark for the general case.

Another promising direction for future work seems to be the connection between CFI-graphs and linear algebra. The CFI problems studied here may give rise to classes of linear equation systems that are solvable in CPT, which would advance the study of expressibility of linear algebra in CPT and thus the connection between CPT and Rank Logic.

On the other hand, our inexpressibility result for CPT without set-like objects separates two of the few known natural fragments of CPT. Our characterisation of set-like objects could provide new insights for other sequence-based formalisms.

References

1 F. Abu Zaid, E. Grädel, M. Grohe, and W. Pakusa. Choiceless Polynomial Time on structures with small Abelian colour classes. In *MFCS 2014*, volume 8634 of *LNCS*, pages 50–62. Springer, 2014.

2 A. Blass. Why sets? In *Pillars of Computer Science*, volume 4800 of *LNCS*, pages 179–198. Springer, 2008. doi:10.1007/978-3-540-78127-1_11.

3 A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100(1):141–187, 1999.

4 A. Blass, Y. Gurevich, and S. Shelah. On polynomial time computation over unordered structures. *J. Symb. Logic*, 67(3):1093–1125, 2002.

5 Andreas Blass, Yuri Gurevich, and Jan Van den Bussche. Abstract state machines and computationally complete query languages. In *International Workshop on Abstract State Machines*, pages 22–33. Springer, 2000.

6 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.

7 A. Chandra and D. Harel. Structure and complexity for relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

8 A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, pages 8–21, 2015.

9 A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *LICS 2009*, pages 113–122, 2009.

10 A. Dawar, D. Richerby, and B. Rossman. Choiceless polynomial time, counting and the Cai–Fürer–Immerman graphs. *Ann. Pure Appl. Logic*, 152(1), 2008.

11 E. Grädel and M. Grohe. Is Polynomial Time Choiceless? In *Fields of Logic and Computation II.*, volume 9300 of *LNCS*, pages 193–209. Springer, 2015.

12 E. Grädel, Ł. Kaiser, W. Pakusa, and S. Schalthöfer. Characterising Choiceless Polynomial Time with First-Order Interpretations. In *LICS*, 2015.

13 E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! *CoRR (a conference version appeared in the proceedings of CSL'15)*, abs/1503.05423, 2015.

14 M. Grohe. The quest for a logic capturing PTIME. In *Logic in Computer Science, 2008, (LICS'08)*, pages 267–271. IEEE, 2008.

15 Y. Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*. Computer Science Press, 1988.

16 N. Immerman. Relational queries computable in polynomial time. *Inf. and Control*, 68:86–104, 1986.

17 B. Laubner. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time*. PhD thesis, HU Berlin, 2011.

18 B. Rossman. Choiceless computation and symmetry. In *Fields of Logic and Computation*, LNCS, pages 565–580. Springer, 2010.

19 M. Y. Vardi. The complexity of relational query languages. In *STOC'82*, pages 137–146. ACM Press, 1982.

Descriptive Complexity of #AC^0 Functions*

Arnaud Durand[1], Anselm Haak[2], Juha Kontinen[3], and Heribert Vollmer[4]

1 **Université Paris Diderot, IMJ-PRG, CNRS UMR 7586, Case 7012, 75205 Paris cedex 13, France**
durand@logique.jussieu.fr
2 **Theoretische Informatik, Leibniz Universität Hannover, Germany**
haak@thi.uni-hannover.de
3 **Department of Mathematics and Statistics, University of Helsinki, Finland**
juha.kontinen@helsinki.fi
4 **Theoretische Informatik, Leibniz Universität Hannover, Germany**
vollmer@thi.uni-hannover.de

Abstract

We introduce a new framework for a descriptive complexity approach to arithmetic computations. We define a hierarchy of classes based on the idea of counting assignments to free function variables in first-order formulae. We completely determine the inclusion structure and show that #P and #AC^0 appear as classes of this hierarchy. In this way, we unconditionally place #AC^0 properly in a strict hierarchy of arithmetic classes within #P. We compare our classes with a hierarchy within #P defined in a model-theoretic way by Saluja et al. We argue that our approach is better suited to study arithmetic circuit classes such as #AC^0 which can be descriptively characterized as a class in our framework.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases finite model theory, Fagin's theorem, arithmetic circuits, counting classes, Skolem function

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.20

1 Introduction

The complexity of arithmetic computations is a current focal topic in complexity theory. Most prominent is Valiant's class #P of all functions that count accepting paths of nondeterministic polynomial-time Turing machines. This class has interesting complete problems like counting the number of satisfying assignments of propositional formulae or counting the number of perfect matchings of bipartite graphs (the so-called permanent [15]).

The class #P has been characterized in a model-theoretic way by Saluja, Subrahmanyam and Thakur in [13]. Their characterization is a natural generalization of Fagin's Theorem: Given a first-order formula with a free relational variable, instead of asking if there exists an assignment to this variable that makes the formula true (NP = ESO), we now ask to count how many such assignments there are. In this way, the class #P is characterized: #P = #FO$^{\text{rel}}$. We use the superscript rel to denote that we are counting assignments to relational variables.

* Partially supported by DFG VO 630/8-1, grant 292767 of the Academy of Finland and grant ANR-14-CE25-0017-02 AGREG of the ANR.

© Arnaud Durand, Anselm Haak, Juha Kontinen, and Heribert Vollmer;
licensed under Creative Commons License CC-BY

From another point of view, the class #P can be seen as the class of those functions that can be computed by arithmetic circuits of polynomial size, i.e., circuits with plus and times gates instead of the usual Boolean gates (cf., e.g., [16]). This is why here we speak of *arithmetic computations*. In the following, all circuit complexity classes we are referring to will be FO-uniform classes, which means that there are FO-formulae describing the circuits for all input lengths (a formal definition will be given).

It is very natural to restrict the resource bounds of such arithmetic circuits. An important class defined in this way is the class #AC^0 of all functions computed by polynomial-size bounded-depth arithmetic circuits. It is interesting to note that #AC^0 and all analogous classes defined by arithmetic circuits, i.e., plus-times circuits, can also be defined making use of a suitable counting process: A witness that a Boolean circuit accepts its input is a so-called proof tree of the circuit, i.e., a minimal subtree of the circuit unwound into a tree, in which all gates evaluate to 1. Then the arithmetic class #AC^0, restricted to binary inputs, can be characterized as the counting class of all functions that count proof trees of (Boolean) AC^0 circuits. The correspondence between arithmetic computations and counting classes is explored in [3]. In this paper, we are mainly interested in these counting classes, and without further mention we use the notation #AC^0 in this vein.

There was no model-theoretic characterization of #AC^0, until it was recently shown in [10] that #AC^0 = #Π_1^{sk}, where #Π_1^{sk} means counting of possible Skolem functions for FO-formulae.

The aim of this paper is to compare the above two model-theoretic characterizations in order to get a unified view for both arithmetic circuit classes, #AC^0 and #P. This is done by noticing that the number of Skolem functions of an FO-formula can be counted as satisfying assignments to free function variables in a Π_1-formula. This gives rise to the idea to restate the result by Saluja et al counting functions instead of relations. We call our class where we count assignments to function variables #FO, in contrast to Saluja et al.'s #FO$^{\mathrm{rel}}$. In this setting, we get #P = #FO = #Π_1, which places both classes within #Π_1.

Furthermore, we show that #AC^0 actually corresponds to a syntactic fragment #Π_1^{prefix} of #Π_1 and, considering further syntactic subclasses of #FO defined by quantifier alternations, we get the inclusions

$$\#\Sigma_0 \subsetneq \#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}} \subsetneq \#\Pi_1 = \#\mathrm{FO} = \#\mathrm{P} \qquad (1)$$
$$\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1$$

Thus we establish (unconditionally, i.e., under no complexity theoretic assumptions) the complete structure of the alternation hierarchy within #FO and show where #AC^0 is located in this hierarchy.

Once we know that only universal quantifiers suffice to obtain the full class, i.e., #Π_1 = #P, it is a natural question to ask how many universal quantifiers are needed to express certain functions. We obtain the result that the hierarchy based on the number of universal variables is infinite; however, a possible connection to the depth hierarchy within #AC^0 remains open.

We see that counting assignments to free function variables instead of relation variables in first-order formulae leads us to a hierarchy of arithmetic classes suitable for a study of the power and complexity of the class #AC^0. The hierarchy introduced by Saluja et al. [13] does not seem suitable for such a goal.

This paper is organized as follows: In the next section, we introduce relevant concepts from finite model theory. Here, we also introduce the Saluja et al. hierarchy, and we explain the model-theoretic characterization of #AC^0. In Sect. 3 we introduce our new framework and the class #FO and its subclasses. In Sect. 4 we determine the full structure of the

alternation hierarchy within #FO and place $\#AC^0$ in this hierarchy, while in Sect. 5 we study the hierarchy defined by the number of universal variables in the $\#\Pi_1$-fragment. In Sect. 6 we turn to the hierarchy defined by Saluja et al. and show that the arithmetic class $\#AC^0$ is incomparable to all except the level-0 class and the full class of this hierarchy. Finally, we conclude in Sect. 7 with some open questions.

Our proofs make use of a number of different results and techniques, some stemming from computational complexity theory (such as separation of Boolen circuit classes or the time hierarchy theorem for nondeterministic RAMs), some from model theory (like closure of certain fragments of first-order logic under extensions or taking substructures) or descriptive complexity (correspondence between time-bounded NRAMs and fragments of existential second-order logic). Most techniques have to be adapted to work in our very low complexity setting (new counting reductions, use of the right set of built-in relations, etc.). Our paper sits right in the intersection of finite model theory and computational complexity theory.

2 Definitions and Preliminaries

In this paper we consider finite σ-structures where σ is a finite vocabulary consisting of relation and constant symbols. For a structure $\mathcal{A}$, $\mathrm{dom}(\mathcal{A})$ denotes its universe. We will always use structures with universe $\{0, 1, \ldots, n-1\}$ for some $n \in \mathbb{N} \setminus \{0\}$. Sometimes we will assume that our structures contain certain *built-in relations and constants*, e.g., $\leq^2$, SUCC^2, BIT^2 and min. In the following, we will always make it clear what built-in relations we allow. The interpretations of built-in symbols are fixed for any size of the universe as follows: $\leq^2$ is the $\leq$-relation on $\mathbb{N}$, min is 0, $\mathrm{SUCC}(i, j)$ is true, iff $i + 1 = j$, and $\mathrm{BIT}(i, j)$ is true, iff the i'th bit of the binary represention of j is 1. We will generally write $\mathrm{enc}_\sigma(\mathcal{A})$ for the binary encoding of a σ-structure $\mathcal{A}$. For this we assume the standard encoding (see e.g. [12]): Relations are encoded row by row by listing their truth values as 0's and 1's. Constants are encoded by the binary representation of their value and thus a string of length $\lceil \log_2(n) \rceil$. A whole structure is encoded by the concatenation of the encodings of its relations and constants except for the built-in numerical predicates and constants: These are not encoded, because they are fixed for any input length.

Since we want to talk about languages accepted by Boolean circuits, we will need the vocabulary

$$\tau_{\mathrm{string}} = (\leq^2, S^1)$$

of binary strings. A binary string is represented as a structure over this vocabulary as follows: Let $w \in \{0,1\}^*$ with $|w| = n$. Then the structure representing this string is the structure with universe $\{0, \ldots, n-1\}$, $\leq^2$ interpreted as the $\leq$-relation on the natural numbers and $x \in S$, iff the x'th bit of w is 1. The structure corresponding to string w is denoted by $\mathcal{A}_w$.

For any k, the fragments Σ_k and Π_k of FO are the classes of all formulae in prenex normal form with a quantifier prefix with k alternations starting with an existential or an universal quantifier, respectively.

In order to define uniformity of circuit families we need FO-interpretations, which are mappings between structures over different vocabularies.

▶ **Definition 1.** Let σ, τ be vocabularies, $\tau = (R_1^{a_1}, \ldots, R_r^{a_r})$. A first-order interpretation (or FO-interpretation)

$$I : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$$

is given by a tuple of FO-formulae $\varphi_0, \varphi_1, \ldots, \varphi_r$ over the vocabulary σ. For some k, φ_0 has k free variables and φ_i has $k \cdot a_i$ free variables for all $i \geq 1$. For each structure $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, these formulae define the structure

$$I(\mathcal{A}) = \langle |I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})} \rangle \in \mathrm{STRUC}[\tau],$$

where the universe is defined by φ_0 and the relations by $\varphi_1, \ldots, \varphi_r$ in the following way:

$$|I(\mathcal{A})| = \{\langle b^1, \ldots, b^k \rangle \mid \mathcal{A} \vDash \varphi_0(b^1, \ldots, b^k)\}$$

$$R_i^{I(\mathcal{A})} = \{(\langle b_1^1, \ldots, b_1^k \rangle, \ldots, \langle b_{a_i}^1, \ldots, b_{a_i}^k \rangle) \in |I(\mathcal{A})|^{a_i} \mid \mathcal{A} \vDash \varphi_i(b_1^1, \ldots, b_{a_i}^k)\}$$

We will now define the class #P and a model-theoretic framework in which the class can be characterized. Here, we follow [13] only changing the name slightly to emphasize that we are counting relations in this setting.

▶ **Definition 2.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in #P, if there is a non-deterministic Turing-machine M such that for all inputs $x \in \{0,1\}^*$,

$$f(x) = \text{number of accepting computation paths of } M \text{ on input } x.$$

▶ **Definition 3.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in $\#\mathrm{FO}^{\mathrm{rel}}$, if there is a vocabulary σ including built-in linear order $\leq$, and an FO-formula $\varphi(R_1, \ldots, R_k, x_1, \ldots, x_\ell)$ over σ with free relation variables $R_1, \ldots, R_k$ and free individual variables $x_1, \ldots, x_\ell$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(S_1, \ldots, S_k, c_1, \ldots, c_\ell) \mid \mathcal{A} \vDash \varphi(S_1, \ldots, S_k, c_1, \ldots, c_\ell\}|.$$

If the input of f is not of this form, we assume f takes the value 0.

In the same fashion we define counting classes using fragments of FO, such as $\#\Sigma_{\mathrm{i}}^{\mathrm{rel}}$ and $\#\Pi_{\mathrm{i}}^{\mathrm{rel}}$ for arbitrary i. In [13] the following was shown for these classes (assuming order as the only built-in relation):

▶ **Theorem 4.** $\#\Sigma_0^{\mathrm{rel}} = \#\Pi_0^{\mathrm{rel}} \subset \#\Sigma_1^{\mathrm{rel}} \subset \#\Pi_1^{\mathrm{rel}} \subset \#\Sigma_2^{\mathrm{rel}} \subset \#\Pi_2^{\mathrm{rel}} = \#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{P}$.

Besides this theorem, it was also shown that the functions in $\#\Sigma_0^{\mathrm{rel}}$ can be computed in polynomial time.

To illustrate the definition just given, we repeat an example from Saluja et al. [13] that will also be important for us later.

▶ **Example 5.** We will show that #3DNF, the problem of counting the number of satisfying assignments of a propositional formula in disjunctive normal-form with at most 3 literals per disjunct, is in the class $\#\Sigma_1^{\mathrm{rel}}$. To do so, we use the vocabulary $\sigma_{\#3\mathrm{DNF}} = (D_0, D_1, D_2, D_3)$. Given a 3DNF-formula φ over variables V, we construct a corresponding σ-structure $\mathcal{A}_\varphi$ with universe V such that for any $x_1, x_2, x_3 \in V$, $D_i(x_1, x_2, x_3)$ holds iff $\bigwedge_{1 \leq j \leq i} \neg x_i \wedge \bigwedge_{i < j \leq 3} x_i$ appears as a disjunct. Now consider the following σ-formula with free relational variable T:

$$\Phi_{\#3\mathrm{DNF}}(T) = \exists x \exists y \exists z \quad \Big(\begin{array}{l} \big(D_0(x,y,z) \wedge T(x) \wedge T(y) \wedge T(z)\big) \\ \vee \big(D_1(x,y,z) \wedge \neg T(x) \wedge T(y) \wedge T(z)\big) \\ \vee \big(D_2(x,y,z) \wedge \neg T(x) \wedge \neg T(y) \wedge T(z)\big) \\ \vee \big(D_3(x,y,z) \wedge \neg T(x) \wedge \neg T(y) \wedge \neg T(z)\big) \Big) \end{array}$$

Observe that $\Phi_{\#3\mathrm{DNF}}$ is a Σ_1-formula. Evaluated on an input structure $\mathcal{A}_\varphi$, it expresses that an assignment to T defines a satisfying assignment of φ. Hence, the number of assignments $\mathbf{T}$ such that $\mathcal{A}_\varphi \models \Phi_{\#3\mathrm{DNF}}(\mathbf{T})$ is equal to the number of satisfying assignments of φ.

We will next recall the definition of Boolean circuits and counting classes defined using them. A circuit is a directed acyclic graph (dag), whose nodes (also called gates) are marked with either a Boolean function (in our case $\wedge$ or $\vee$), a constant (0 or 1), or a (possibly negated) bit of the input. Also, one gate is marked as the output gate. On any input, a circuit computes a Boolean function by evaluating all gates according to what they are marked with. The value of the output gate then is the function value for that input.

When we want circuits to work on different input lengths, we have to consider families of circuits: A family contains a circuit for any input length $n \in \mathbb{N}$. Families of circuits allow us to talk about languages being accepted by circuits: A circuit family $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ is said to accept (or decide) the language L, if it computes its characteristic function c_L:

$$C_{|x|}(x) = c_L(x) \text{ for all } x.$$

The complexity classes in circuit complexity are classes of languages that can be decided by circuit families with certain restrictions to their depth and size. The depth here is the length of a longest path from any input gate to the output gate of a circuit and the size is the number of non-input gates in a circuit. Depth and size of a circuit family are defined as functions accordingly.
Above, we have not restricted the computability of the circuit $C_{|x|}$ from x in any way. This is called non-uniformity, which allows such circuit families to even compute non-recursive functions. Since we want to stay within #P, we need some notion of uniformity. For this, we first define the vocabulary for Boolean circuits as FO-structures:

$$\tau_{\text{circ}} = (E^2, G^1_\wedge, G^1_\vee, B^1, r^1),$$

where the relations are interpreted as follows:

- $E(x, y)$: y is a child of x
- $G_\wedge(x)$: gate x is an and-gate
- $G_\vee(x)$: gate x is an or-gate
- $B(x)$: gate x is a true leaf of the circuit
- $r(x)$: x is the root of the circuit

We will now define FO-uniformity of Boolean circuits in general and the class AC^0.

▶ **Definition 6.** A circuit family $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ is said to be first-order uniform (FO-uniform) if there is an FO-interpretation

$$I : \text{STRUC}[\tau_{\text{string}} \cup (\text{BIT}^2)] \to \text{STRUC}[\tau_{\text{circ}}]$$

mapping any structure $\mathcal{A}_w$ over τ_{string} to the circuit $C_{|w|}$ given as a structure over the vocabulary τ_{circ}.

Note that by [4] this uniformity coincides with the maybe better known DLOGTIME-uniformity for many familiar circuit classes (and in particular for all classes studied in this paper).

▶ **Definition 7.** A language $L \subseteq \{0, 1\}^*$ is in AC^0, if there is an FO-uniform circuit family with constant depth and polynomial size accepting L.

It is known that the just given class coincides with the class FO of all languages definable in first-order logic [5, 12], i.e., informally: $\text{AC}^0 = \text{FO}$. This identity holds if our logical language includes the built-in relations of linear order and BIT. Though it is known that

linear order can be defined using BIT, we require that both are present in our language, because we consider very restricted quantifier prefixes where $\leq$ cannot be defined with BIT.

We will next define counting classes corresponding to Boolean circuit families. For a nondeterministic Turing machine, the witnesses we want to count are the accepting paths of the machine on a given input. Considering polynomial time computations, this concept gives rise to the class #P. A witness that a Boolean circuit accepts its input is a so-called *proof tree*, a minimal subtree of the circuit showing that it evaluates to true for a given input. For this, we first unfold the given circuit into tree shape, and we further require that it is in *negation normal form* (meaning that negations only occur directly in front of literals). A proof tree then is a subtree we get by choosing for any $\vee$-gate exactly one child and for any $\wedge$-gate all children, such that every leaf which we reach in this way is a true literal. This allows us to define the class #AC^0 as follows:

▶ **Definition 8.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in #AC^0, if there is an FO-uniform circuit family $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ such that for all $w \in \{0,1\}^*$,

$$f(w) = \text{number of proof trees of } C_{|w|}(w).$$

As was shown in [10], there is a model-theoretic characterization of #AC^0. For this, let us define the Skolemization of an FO-formula φ in prenex normal form by removing all existential quantifiers and replacing each existentially quantified variable in the quantifier-free part of φ by a term consisting of a function application to those variables quantified universally to the left of the original existential quantifier. In other words, every existential variable is replaced by its so-called *Skolem function*. Now, #AC^0 contains exactly those functions that can be given as the number of Skolem functions for a given FO-formula.

▶ **Definition 9.** A function $f\colon \{0,1\}^* \to \mathbb{N}$ is in the class $\#\Pi_1^{\mathrm{sk}}$ if there is a vocabulary σ including built-in $\leq$, BIT and min and a first-order sentence φ over σ in prenex normal form

$$\varphi \triangleq \exists y_1 \forall z_1 \exists y_2 \forall z_2 \ldots \exists y_{k-1} \forall z_{k-1} \exists y_k \; \psi(\overline{y}, \overline{z}),$$

where ψ is quantifier-free such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, $f(\mathrm{enc}_\sigma(\mathcal{A}))$ is equal to the number of tuples $(f_1, \ldots, f_k)$ of functions such that

$$\mathcal{A} \models \forall z_1 \ldots \forall z_{k-1} \; \psi(f_1, f_2(z_1), \ldots, f_k(z_1, \ldots, z_{k-1}), z_1, \ldots, z_{k-1}).$$

If the input of f is not of this form, we assume f takes the value 0.

This means that $\#\Pi_1^{\mathrm{sk}}$ contains those functions that, for a fixed FO-formula φ over some vocabulary σ, map an input w to the number of Skolem functions for φ on $\mathcal{A} = \mathrm{enc}_\sigma^{-1}(w)$.

▶ **Theorem 10.** $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{sk}}$.

The above mentioned result $\mathrm{FO} = \mathrm{AC}^0$ [5, 12] requires built-in order and BIT; hence it is no surprise that also for the theorem just given these relations are needed, and this is the reason why they also appear in Def. 9.

3 Connecting the Characterizations of #AC^0 and #P

We will now establish a unified view on the model-theoretic characterizations of both #AC^0 and #P. This will be done by viewing #AC^0 as a syntactic subclass of #FO. In Theorem 10 we characterized #AC^0 by a process of counting assignments to function variables in

FO-formulae, but only in a very restricted setting. It is natural to define the process of counting functions in a more general way, similar to the framework of [13], repeated here in Def. 3, where Saluja et al. count assignments to free relation variables in FO-formulae to obtain their characterization of #P.

▶ **Definition 11.** #FO is the class of all functions $f\colon \{0,1\}^* \to \mathbb{N}$ for which there is a vocabulary σ, including built-in $\leq$, BIT and min, and an FO-formula $\varphi(F_1,\dots,F_k,x_1,\dots,x_\ell)$ over σ with free function variables $F_1,\dots,F_k$ and free individual variables $x_1,\dots,x_\ell$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = \left|\left\{(f_1,\dots,f_k,c_1,\dots,c_\ell) \mid \mathcal{A} \vDash \varphi(f_1,\dots,f_k,c_1,\dots,c_\ell)\right\}\right|.$$

If the input of f is not of this form, we assume f takes the value 0.

In the same fashion we define counting classes using fragments of FO, such as $\#\Sigma_\mathrm{i}$ and $\#\Pi_\mathrm{i}$ for arbitrary i. Note, that the free individual variables could also be seen as free function variables of arity 0.

We stress that our signatures in the above definition include symbols $\leq$, BIT, and min with their standard interpretations; as argued already several times, these built-ins are necessary in order to obtain a close correspondence between standard circuit classes like AC^0, TC^0 and first-order logic (but cf. results that consider weaker logics and relate them to presumably smaller non-standard complexity classes [6]). In contrast to our definition, Saluja et al. (see Def. 3) only use built-in order. Still, we will now see that both concepts, counting relations and counting function, are in fact equivalent as long as we use all of FO, even with different sets of built-in relations.

▶ **Theorem 12.** $\#\mathrm{FO}^{\mathrm{rel}} = \#\mathrm{FO} = \#\mathrm{P}$.

Proof. The inclusion $\#\mathrm{FO}^{\mathrm{rel}} \subseteq \#\mathrm{FO}$ is shown as follows: Let $f \in \#\mathrm{FO}^{\mathrm{rel}}$ via the formula φ containing free relation variables $R_1,\dots,R_k$. We can replace R_i by a function variable F_i of the same arity for all i. We then add a conjunct to the formula ensuring that for these functions only min and the element $x > \min$ with $\forall y(y < x \to y = \min)$ are allowed as function values. Then each occurrence of $R_i(\overline{z})$ can be replaced by $F_i(\overline{z}) = \min$.

The inclusion $\#\mathrm{FO} \subseteq \#\mathrm{P}$ is straightforward. The inclusion $\#\mathrm{P} \subseteq \#\mathrm{FO}^{\mathrm{rel}}$ was shown in [13]. ◀

Note that $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{sk}}$ does not directly arise from this definition by choosing an appropriate fragment of FO because of the restricted usage of the second-order variables in Def. 9. Still, we will characterize $\#\mathrm{AC}^0$ as a syntactic subclass of #FO as follows.

▶ **Definition 13.** Let $\#\Pi_1^{\mathrm{prefix}}$ be the class of all functions f for which there is a Π_1-formula $\varphi(\overline{G},\overline{x}) = \forall y_1 \dots \forall y_k \psi(\overline{G},\overline{x},y_1,\dots,y_k)$ over some vocabulary σ, where ψ is quantifier-free and in which all arity-a functions G (for any a) occur in ψ only as $G(y_1,\dots,y_a)$ such that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(\overline{g},\overline{c}) \mid \mathcal{A} \vDash \varphi(\overline{g},\overline{c})\}|.$$

If the input of f is not of this form, we assume f takes the value 0.

▶ **Lemma 14.** $\#\mathrm{AC}^0 = \#\Pi_1^{\mathrm{prefix}}$.

Proof. By Theorem 10 it suffices to show $\#\Pi_1^{\mathrm{sk}} = \#\Pi_1^{\mathrm{prefix}}$. We consider first the inclusion $\#\Pi_1^{\mathrm{sk}} \subseteq \#\Pi_1^{\mathrm{prefix}}$. Let $g \in \#\Pi_1^{\mathrm{sk}}$ via a formula φ as in Def. 9. Then we can simply replace

the occurrences of variables y_i in ψ by the corresponding function terms. The resulting formula is prefix-restricted as needed and directly shows $g \in \#\Pi_1^{\text{prefix}}$.

For $\#\Pi_1^{\text{prefix}} \subseteq \#\Pi_1^{\text{sk}}$, let $g \in \#\Pi_1^{\text{prefix}}$ via a formula φ. Since all function symbols occurring in φ are only applied to a unique prefix of the universally quantified variables, they can be seen as Skolem functions of suitable existentially quantified variables. Thus, we can replace the occurrences of the function symbols by new variables that are existentially quantified at adequate positions between the universally quantified variables. If for example, the input for a function was $x_1, \dots, x_\ell$, then the new variable is quantified after the part $\forall x_1 \dots \forall x_\ell$ of the quantifier prefix. Strict alternations in the quantifier-prefix, as required for $\#\Pi_1^{\text{sk}}$, can be achieved by adequately adding dummy-variables in between and forcing them to be equal to min. This yields a formula φ' that shows $g \in \#\Pi_1^{\text{sk}}$. ◀

4 An Alternation Hierarchy in #FO

In this section we study a hierarchy within #FO based on quantifier alternations. Interestingly, our approach allows us to locate #AC^0 in this hierarchy. First we note that the whole hierarchy collapses to a quite low class.

▶ **Theorem 15.** $\#\text{FO} = \#\Pi_1$.

Proof. Let $h \in \#\text{FO}$ via an FO-formula $\varphi(\overline{f}, \overline{x})$ in prenex normal form. We show how to transform φ to a Π_1-formula also defining h. As a first step, we change φ in such a way that for each existential variable instead of "there is an x" we say "there is a smallest x". Formally, this can be done with the following transformation:

$$\exists x \theta(x) \rightsquigarrow \exists x(\theta(x) \wedge \forall z(\neg\theta(z) \vee x < z \vee x = z))$$

applied recursively to all existential quantifiers in φ. Note that now for every satisfied $\exists$-quantifier there is exactly one witness.

For the sake of argument, suppose that after the above transformation and re-conversion to prenex normal form the formula $\varphi(\overline{f}, \overline{x})$ corresponds to

$$\varphi'(\overline{f}, \overline{x}) = \exists z_1 \forall y_1 \exists z_2 \dots \forall y_{\ell-1} \exists z_\ell \psi(\overline{f}, \overline{x}, z_1, \dots, z_\ell, y_1, \dots, y_{\ell-1})$$

where ψ is quantifier-free. Looking at the Skolemization of φ', our transformation ensures that every existentially quantified variable has a unique Skolem function. Thus,

$$\varphi''(\overline{f}, \overline{x}, g_1, \dots, g_\ell) = \forall y_1 \dots \forall y_{\ell-1} \psi(\overline{f}, \overline{x}, g_1, g_2(y_1), \dots, g_\ell(y_1, \dots, y_{\ell-1}), y_1, \dots, y_{\ell-1})$$

shows $h \in \#\Pi_1$. ◀

Next we look at the lowest class in our hierarchy and separate it from #AC^0.

▶ **Theorem 16.** $\#\Sigma_0 \subsetneq \#\text{AC}^0$.

Proof. We start by showing the inclusion. Certain observations in that proof will then almost directly yield the strictness. Let $f \in \#\Sigma_0$ via the quantifier-free FO-formula $\varphi(F_1, \dots, F_k, x_1, \dots, x_\ell)$ over some vocabulary σ, where $F_1, \dots, F_k$ are free function variables and $x_1, \dots, x_\ell$ are free individual variables, that is,

$$f(\text{enc}_\sigma(\mathcal{A})) = |\{(f_1, \dots, f_k, c_1, \dots, c_\ell) \mid \mathcal{A} \vDash \varphi(f_1, \dots, f_k, c_1, \dots, c_\ell)\}|.$$

Without loss of generality we can assume that in φ no nesting of functions occurs. If there is an occurrence of a function G as an argument for function H, then we can replace the

occurrence of G by a new free variable and force this variable to be equal to the function value. This ensures that there is only one unique assignment to this new free variable.

Let $A := \mathrm{dom}(\mathcal{A})$. For all i, let a_i be the arity of F_i and let m_i be the number of syntactically different terms that occur as inputs to F_i within φ. Let $e_{i1}, \dots, e_{im_i}$ be those terms in the order of their occurrence within φ and let $\varphi'(y_{11}, \dots, y_{1m_1}, \dots, y_{k1}, \dots, y_{km_k}, x_1, \dots, x_\ell)$ be φ after replacing for all i, j all occurrences of $F_i(e_{ij})$ by the new free variable y_{ij}. Let $m := \sum_i m_i$.

Considering a fixed assignment to the variables $x_1, \dots, x_\ell$, each e_{ij} has a fixed value. Thus, we can use free individual variables in order to count the number of assignments to all terms $F_i(e_{ij})$ for all (i, j). After that, all f_i have to be chosen in accordance with those choices to get the correct number of functions that satisfy the formula. Formally, this is done as follows:

$$\begin{aligned} f(\mathrm{enc}_\sigma(\mathcal{A})) &= \sum_{\overline{c} \in A^\ell} \sum_{\substack{(f_1,\dots,f_k)\in \\ A^{A^{a_1}} \times \cdots \times A^{A^{a_k}}}} [\mathcal{A} \vDash \varphi(f_1, \dots, f_k, c_1, \dots, c_\ell)] \\ &= \sum_{\overline{c} \in A^\ell} \sum_{\overline{d} \in A^m} \sum_{(f_1,\dots,f_k)\in G} [\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})], \end{aligned}$$

where $G := \{(f_1, \dots, f_k) \in A^{A^{a_1}} \times \cdots \times A^{A^{a_k}} \mid \forall (i, j) :\ \mathcal{A} \vDash\ d_{ij} = f_i(e_{ij})\}$.

Since $[\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})]$ does not depend on $(f_1, \dots, f_k)$, we can multiply by the cardinality of G instead of summing:

$$f(\mathrm{enc}_\sigma(\mathcal{A})) = \sum_{\substack{\overline{c} \in A^\ell, \\ \overline{d} \in A^m}} [\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})] \cdot |G|$$

Now we are in a position to show $f \in \#\mathrm{AC}^0$.

The sum only has polynomially many summands and thus is obviously possible in $\#\mathrm{AC}^0$.

For $[\mathcal{A} \vDash \varphi'(\overline{d}, \overline{c})]$, the circuit only has to evaluate a quantifier-free formula depending on an assignment that is given by the path from the root to the current gate. This is similar to the corresponding part of the proof of $\mathrm{FO} = \mathrm{AC}^0$ and thus can be done in $\mathrm{AC}^0 \subseteq \#\mathrm{AC}^0$.

For $|G|$ we first note that the total number of possible assignments for $\overline{f}$ is

$$|A^{A^{a_1}} \times \cdots \times A^{A^{a_k}}| = |A|^{\sum_i |A|^{a_i}}.$$

The definition of G fixes for each function f_i the function value on at most m_i inputs to be equal to some d_{ij}. This means, that the function value on at least $|A|^{a_i} - m_i$ inputs is not determined by the definition of G and can thus be freely chosen.

If for some (i, j), e_{ij} is semantically equal to $e_{ij'}$ for some $j' < j$, it has to hold that $d_{ij} = d_{ij'}$. Additionally, this reduces the amount of function values that are fixed by the d_{ij} by 1. To make this formal we define for any (i, j)

$$S_{ij} = \{j' \mid j' < j \text{ and } \mathcal{A} \vDash\ e_{ij} = e_{ij'}\}.$$

From the above considerations we get

$$|G| = [\bigwedge_{(i,j)} \bigwedge_{j'} (j' \in S_{ij}) \to d_{ij} = d_{ij'}] \cdot |A|^{\sum_i |A|^{a_i} - \sum_i m_i} \cdot |A|^{\sum_{ij} [S_{ij} \neq \emptyset]}.$$

Since the a_i and m_i are constants and S_{ij} is FO-definable, $|G|$ can be computed in $\#\mathrm{AC}^0$. This concludes the proof for $\#\Sigma_0 \subseteq \#\mathrm{AC}^0$.

Note that for any $\#\Sigma_0$-function f defined using a Σ_0-formula without free second-order variables, $f(w)$ is bounded polynomially in $|w|$ for all inputs w. On the other hand, the above proof shows that for any $\#\Sigma_0$-function f defined using a Σ_0-formula with at least one free second-order variable, there are constants $c_i > 0$ such that $f(w)$ divisible by $|w|^{\sum_i |w|^{c_i} - \text{const}}$ for all inputs w. Thus, the function $f(w) = |w|^{\lceil |w|/2 \rceil} \in \#\mathrm{AC}^0$ is not in $\#\Sigma_0$ which means $\#\Sigma_0 \neq \#\mathrm{AC}^0$. ◀

▶ **Theorem 17.** $\#\Pi_1^{\mathrm{sk}} \subsetneq \#\Pi_1$.

Proof. From the above we know that the left class is equal to $\#\mathrm{AC}^0$ and the right class is equal to #P. Strict inclusion now follows immediately from the following considerations: Let $\mathcal{F}$ be a class of functions $\{0,1\}^* \to \mathbb{N}$. Then the class $\mathsf{C} \cdot \mathcal{F}$ is the class of all languages L for which there are $f, g \in \mathcal{F}$ such that for all $x \in \{0,1\}^*$, $x \in L \Leftrightarrow f(x) > g(x)$. In [1] it was shown that $\mathrm{TC}^0 = \mathsf{C} \cdot \#\mathrm{AC}^0$. Also, it is well known that $\mathrm{PP} = \mathsf{C} \cdot \#\mathrm{P}$. Allender's separation $\mathrm{TC}^0 \neq \mathrm{PP}$ [2] now directly yields $\#\mathrm{AC}^0 \neq \#\mathrm{P}$. ◀

So far we have identified the following hierarchy:

$$\#\Sigma_0 \subsetneq \#\Pi_1^{\mathrm{sk}} = \#\mathrm{AC}^0 \subsetneq \#\Pi_1 = \#\mathrm{P}. \tag{2}$$

Next we turn to the class $\#\Sigma_1$ and show that it forms a different branch between $\#\Sigma_0$ and $\#\Pi_1$.

▶ **Lemma 18.** *There exists a function F which is in $\#\Pi_1^{\mathrm{sk}}$ but not in $\#\Sigma_1$.*

Proof. Let $\tau = \{E, c, d, \leq, \mathrm{BIT}, \min\}$ where E is a binary relation symbol and c, d are constant symbols. Let us consider the function F defined by the number of Skolem functions of variable z in the formula $\varphi = \forall x \forall y \exists z\ \psi(x, y, z)$ with

$$\psi = (E(x,y) \to z = c \vee z = d) \wedge (\neg E(x,y) \to z = c).$$

For a given τ-structure $\mathcal{A}$ with $c^{\mathcal{A}} \neq d^{\mathcal{A}}$, it is clear that:

$$F(\mathrm{enc}_\tau(\mathcal{A})) = |\{f \mid \mathcal{A} \models \forall x \forall y\ \psi(x, y, f(x,y))\}| = 2^{|E^{\mathcal{A}}|},$$

since each edge gives rise to two possible values for $z = f(x,y)$ and each non edge to only one value. Thus, $F \in \#\Pi_1^{\mathrm{sk}}$.

Suppose now that $F \in \#\Sigma_1$ i.e. that there exists $\phi(\overline{g}, \overline{x}) \in \Sigma_1$ such that for all τ-structures $\mathcal{G}$,

$$F(\mathrm{enc}_\tau(\mathcal{G})) = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{G} \models \phi(\overline{g}_0, \overline{a})\}|$$

and in particular for $\mathcal{A}$ as above,

$$2^{|E^{\mathcal{A}}|} = F(\mathrm{enc}_\tau(\mathcal{A})) = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{A} \models \phi(\overline{g}_0, \overline{a})\}|.$$

Now consider the following structure $\mathcal{A}'$ defined simply by extending $\mathrm{dom}(\mathcal{A}) = \{0, ..., n-1\}$ by two new elements, i.e., $\mathrm{dom}(\mathcal{A}') = \{0, ..., n+1\}$. Note that $E^{\mathcal{A}} = E^{\mathcal{A}'}$, hence the two structures have the same number of edges. To make the presentation simpler, suppose $\overline{g} = g$ and that the arity of g is one. Any given $g_0 : \mathrm{dom}(\mathcal{A}) \longrightarrow \mathrm{dom}(\mathcal{A})$, can be extended in several ways on the domain $\mathrm{dom}(\mathcal{A}')$ in particular as g_1 and g_2 below:

- $g_1(x) = g_0(x)$ for all $x \in \mathrm{dom}(\mathcal{A})$ and $g_1(n) = c$, $g_1(n+1) = d$.
- $g_2(x) = g_0(x)$ for all $x \in \mathrm{dom}(\mathcal{A})$ and $g_2(n) = d$, $g_2(n+1) = c$.

Formulas in Σ_1 are stable under extension of models so if $\overline{a}$ and g_0 are such that $\mathcal{A} \models \phi(g_0, \overline{a})$ then $\mathcal{A}' \models \phi(g_1, \overline{a})$ and $\mathcal{A}' \models \phi(g_2, \overline{a})$. Hence,

$$|\{(g', \overline{a}) \mid \mathcal{A}' \models \phi(g', \overline{a})\}| > |\{(g_0, \overline{a}) \mid \mathcal{A} \models \phi(g_0, \overline{a})\}|.$$

On the other hand, $F(\mathrm{enc}_\tau(\mathcal{A})) = F(\mathrm{enc}_\tau(\mathcal{A}'))$ holds, hence our assumpion that $\phi(g, \overline{x}) \in \Sigma_1$ defines F has led to a contradiction. ◀

For the opposite direction, we first show the following lemma.

▶ **Lemma 19.** *The function* #3DNF *is complete for* #P *under* AC^0*-Turing-reductions.*

Proof. A reduction of the #P-complete problem #3CNF to #3DNF is as follows: Given a 3CNF-formula φ over n variables, we first construct $\varphi' = \neg\varphi$. This is a 3DNF-formula. Obviously, the number of satisfying assignments of φ is equal to 2^n minus the number of satisfying assignments of φ'. Since this reduction can be computed by an AC^0-circuit and moreover #3CNF is #P-complete under AC^0-reductions (as follows from the standard proof of the NP-completeness of SAT), #3DNF is complete for #P under AC^0-Turing-reductions. ◀

▶ **Lemma 20.** *There exists a function F which is in $\#\Sigma_1$ but not in $\#\Pi_1^{\mathrm{sk}}$.*

Proof. Using FTC^0 to denote the functional version of TC^0, we first note that $\mathrm{FTC}^0 \neq \#\mathrm{P}$: For the sake of contradiction, assume $\mathrm{FTC}^0 = \#\mathrm{P}$. Making use of the complexity-theoretic operator C (see proof of Theorem 17), we obtain $\mathrm{PP} = \mathsf{C} \cdot \#\mathrm{P} \subseteq \mathsf{C} \cdot \mathrm{FTC}^0 = \mathrm{TC}^0$, but this is a contradiction to $\mathrm{TC}^0 \subsetneq \mathrm{PP}$ [2].

We now show this lemma by modifying the counting problem #3DNF to get a #P-complete function inside of $\#\Sigma_1$. If the reduction we use can be computed in FTC^0, the modified version of #3DNF can not be in $\#\Pi_1^{\mathrm{sk}} \subseteq \mathrm{FTC}^0$, because this would contradict $\mathrm{FTC}^0 \neq \#\mathrm{P}$.

Consider the vocabulary $\sigma_{3\mathrm{DNF}}$ and the formula $\Phi_{\#3\mathrm{DNF}}(T)$ from example 5. Let σ be the vocabulary extending $\sigma_{3\mathrm{DNF}}$ with built-in $\leq$, BIT and min. To get a function in $\#\Sigma_1$, we need to use a free function variable instead of the free relation variable T. Since we cannot use universal quantifiers, relations cannot be represented uniquely as functions of the same arity. In order to still get a #P-complete problem, we want to make sure that compared to #3DNF, the function value of our new counting function only differs from the one of #3DNF by a factor depending on the input length, not on the specific satisfying assignments. To achieve this, we encode the relation T as a function F as follows: interpret for all x an even function value $F(x)$ as $T(x)$ being false and an odd function value $F(x)$ as $T(x)$ being true. Thus, the number of 1's and 0's in a satisfying assignment do not influence the factor by which the new counting function differs from #3DNF.

Following this idea we define for all σ-structures $\mathcal{A}$

$$\#3\mathrm{DNF}^{\mathrm{func}}(\mathrm{enc}_\sigma(\mathcal{A})) = |\{f \mid \mathcal{A} \vDash \Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(f)\}|,$$

where $\Phi_{\#3\mathrm{DNF}^{\mathrm{func}}}(F)$ is $\Phi_{\#3\mathrm{DNF}}(T)$ after replacing for all variables x subformulae of the form $T(x)$ by $\mathrm{BIT}(\min, F(x))$. By definition, $\#3\mathrm{DNF}^{\mathrm{func}} \in \#\Sigma_1$.

We now want to reduce #3DNF to $\#3\mathrm{DNF}^{\mathrm{func}}$. Since the idea above only works if the universe has even cardinality, the first step of the reduction is doubling the size of the universe. Let $\mathcal{A}$ be a structure and $\mathcal{A}'$ the structure that arises from $\mathcal{A}$ by doubling the size of the universe. Let $A = \{0, \ldots, n-1\}$ and $A' = \{0, \ldots, 2n-1\}$ be their respective universes.

Each assignment for T with $\mathcal{A} \vDash \Phi_{\#3\text{DNF}}(T)$ gives rise to the following set of assignments for f with $\mathcal{A}' \vDash \Phi_{\#3\text{DNF}^{\text{func}}}(f)$:

$$S_T = \{f \colon A' \to A' \mid \text{for all } x \in A : f(x) \equiv 1 \mod 2 \quad \Leftrightarrow \quad T(x)\}.$$

These sets are disjoint and by definition of $\Phi_{\#3\text{DNF}^{\text{func}}}(f)$ their union is equal to $\{f \mid \mathcal{A}' \vDash \Phi_{\#3\text{DNF}^{\text{func}}}(f)\}$. For each T, the functions f in S_T have n choices for $f(x)$, if $x \in A$ and $2n$ choices, if $x \notin A$. Thus, $|S_T| = |A|^{|A|} \cdot (2 \cdot |A|)^{|A|}$, yielding

$$\#3\text{DNF}(\text{enc}_{\sigma_{3\text{DNF}}}(\mathcal{A})) = \frac{\#3\text{DNF}^{\text{func}}(\text{enc}_\sigma(\mathcal{A}'))}{|A|^{2|A|} \cdot 2^{|A|}}.$$

Doubling the size of the universe can be done in FTC^0 by adding the adequate number of 0-entries in the encodings of all relations.

The term $|A|^{2|A|} \cdot 2^{|A|}$ can be computed in $\#\text{AC}^0 \subseteq \text{FTC}^0$ and division can be done in FTC^0 due to [11].

Since #3DNF is #P-complete under AC^0-Turing-reductions by Lemma 19, this means that $\#3\text{DNF}^{\text{func}}$ is #P-complete under TC^0-Turing-reductions. ◀

So Lemmas 18 and 20 show that $\#\Sigma_1$ and $\#\Pi_1^{\text{sk}}$ are incomparable, and we obtain the inclusion chain $\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 = \#\text{P}$. Together with (2) we therefore obtain

$$\begin{array}{ccccc} & & \#\text{AC}^0 = \#\Pi_1^{\text{prefix}} & & \\ \#\Sigma_0 & \subsetneq & \cap & \subsetneq & \#\Pi_1 = \#\text{FO} = \#\text{P} \\ & & \#\Sigma_1 & & \end{array} \tag{1}$$

5 Hierarchy Based on the Number of Universal Variables

In this section we study another hierarchy in #FO based on syntactic restrictions, this time given by the number of universal variables.

Let Π_1^k denote the class of Π_1 formulae of the form

$$\forall x_1 \cdots \forall x_m \psi,$$

where $m \leq k$ and ψ is a quantifier-free formula. The function class corresponding to Π_1^k is denoted by $\#\Pi_1^k$. We will show that

$$\#\Pi_1^k \subsetneq \#\Pi_1^{k+1}, \tag{3}$$

for all $k \geq 1$. These results can be shown by applying a result of Grandjean and Olive which we will discuss next.

▶ **Definition 21.** We denote by $\text{ESO}_f(k\forall)$ the class of ESO-sentences in Skolem normal form

$$\exists f_1 \ldots \exists f_n \forall x_1 \ldots \forall x_r \psi,$$

where $r \leq k$, and ψ is a quantifier-free formula.

It was shown in [9] that with respect to any finite signature σ

$$\text{ESO}_f(k\forall) = \text{NTIME}_{\text{RAM}}(n^k),$$

where $\text{NTIME}_{\text{RAM}}(n^k)$ denotes the family of classes of σ-structures that can be recognized by a non-deterministic RAM in time $O(n^k)$. Note that by [8],

$$\text{NTIME}_{\text{RAM}}(n^k) \subsetneq \text{NTIME}_{\text{RAM}}(n^{k+1}).$$

These results can be used to show the strictness of the variables hierarchy (see (3)).

▶ **Theorem 22.** *Let $k \geq 1$. Then*

$$\#\Pi_1^k \subsetneq \#\Pi_1^{k+1}.$$

Proof. Let us fix $\sigma = \{<, \mathrm{BIT}, \min, P\}$, where P is unary. By the above there exists a sentence $\exists f_1 \cdots \exists f_n \psi \in \mathrm{ESO}_f(k+1\forall)[\sigma]$ defining a binary language L which cannot be defined by any sentence $\chi \in \mathrm{ESO}_f(k\forall)[\sigma]$. We claim that the function F associated with the formula $\psi(f_1, ..., f_n) \in \Pi_1^{k+1}$,

$$F(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(f_1, ..., f_n) \mid \mathcal{A} \models \psi(f_1, ..., f_n)\}|,$$

is not a member of $\#\Pi_1^k$. For a contradiction, assume that $F \in \#\Pi_1^k$. Then there exists a formula $\chi(\overline{g}, \overline{y}) \in \Pi_1^k$ such that

$$F(\mathrm{enc}_\sigma(\mathcal{A})) = |\{(\overline{g}, \overline{y}) \mid \mathcal{A} \models \chi(\overline{g}, \overline{y})\}|$$

By the above, the sentence $\exists \overline{g} \exists \overline{y} \chi$ defines the language L, and hence contradicts the assumption that L cannot be defined by any $\mathrm{ESO}_f(k\forall)[\sigma]$-sentence. ◀

It is an interesting open question to study the relationship of $\#\mathrm{AC}^0$ with the classes $\#\Pi_1^k$.

6 #AC^0 compared to the classes from Saluja et al.

In this section we study the relationship of $\#\mathrm{AC}^0$ to the syntactic classes introduced in [13]. As in [13], these classes are defined assuming a built-in order relation only.

▶ **Theorem 23.**

- $\#\Sigma_0^{\mathrm{rel}} \subsetneq \#\mathrm{AC}^0$,
- *Let $\mathcal{C} \in \{\#\Sigma_1^{\mathrm{rel}}, \#\Pi_1^{\mathrm{rel}}, \#\Sigma_2^{\mathrm{rel}}\}$. Then the following holds: $\#\mathrm{AC}^0 \not\subseteq \mathcal{C}$ and $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$.*

Proof. The proof of the inclusion $\#\Sigma_0^{\mathrm{rel}} \subsetneq \#\mathrm{AC}^0$ is analogous to the proof of Theorem 16 and is thus omitted.

For the second statement recall that $\#\Sigma_1^{\mathrm{rel}} \subset \#\Pi_1^{\mathrm{rel}} \subset \#\Sigma_2^{\mathrm{rel}}$. The claim $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$ for $\mathcal{C} \in \{\#\Sigma_1^{\mathrm{rel}}, \#\Pi_1^{\mathrm{rel}}, \#\Sigma_2^{\mathrm{rel}}\}$ can be proven as follows: From Example 5 we know that $\#3\mathrm{DNF} \in \mathcal{C}$ and from Lemma 19 we know that #3DNF is #P-complete under AC^0-Turing-reductions. Now suppose $\#3\mathrm{DNF} \in \#\mathrm{AC}^0$. Then $\#\mathrm{P} \subseteq \mathrm{FAC}^{0\#\mathrm{AC}^0} \subseteq \mathrm{FTC}^0$ [10], contradicting $\mathrm{FTC}^0 \neq \#\mathrm{P}$, which was shown in the proof of Lemma 20. Hence $\#3\mathrm{DNF} \notin \#\mathrm{AC}^0$ and $\mathcal{C} \not\subseteq \#\mathrm{AC}^0$.

It remains to show $\#\mathrm{AC}^0 \not\subseteq \mathcal{C}$. We show this by an argument similar to the proof that #HAMILTONIAN is not in $\#\Sigma_2^{\mathrm{rel}}$, showing the separation of $\#\Sigma_2^{\mathrm{rel}}$ from #FO, see Theorem 2 in [13]. We will show that a very simple function f on encodings of τ_{string}-structures is not in $\mathcal{C}$. Define f as follows: $f(w) = 1$, if $|w|$ is even, and $f(w) = 0$ otherwise. Obviously $f \in \#\mathrm{AC}^0$. It now suffices to show that $f \notin \#\Sigma_2^{\mathrm{rel}}$. For contradiction, assume that $f \in \#\Sigma_2^{\mathrm{rel}}$ via a formula $\phi(\overline{R}, \overline{x}) \in \Sigma_2^{\mathrm{rel}}$, where

$$\phi(\overline{R}, \overline{x}) = \exists \overline{u} \forall \overline{v} \theta(\overline{R}, \overline{x}, \overline{u}, \overline{v}),$$

and θ is a quantifier-free formula. Let s and t be the lengths of the tuples $\overline{u}$ and $\overline{x}$, respectively. Let $n \geq s + t + 1$ be even and let $w \in \{0,1\}^n$. By the assumption, there exists $\overline{\mathbf{R}}$, $\overline{\mathbf{x}}$, $\overline{\mathbf{u}}$ such that

$$\mathcal{A}_w \models \forall \overline{v} \theta(\overline{\mathbf{R}}, \overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{v}).$$

By the choice of n, we can find $i \in \{0, .., n-1\}$ such that i does not appear in the tuples $\overline{\mathbf{u}}$ and $\overline{\mathbf{x}}$. Let $\mathcal{A}_{w'}$ denote the structure arising by removing the element i from the structure $\mathcal{A}_w$, and let $\overline{\mathbf{R}}^*$ denote the relations arising by removing tuples with the element i from $\overline{\mathbf{R}}$. By closure under substructures of universal first-order formulae, it follows that

$$\mathcal{A}_{w'} \models \forall \overline{v} \theta(\overline{\mathbf{R}}^*, \overline{\mathbf{x}}, \overline{\mathbf{u}}, \overline{v}),$$

implying that $f(\mathcal{A}_{w'}) \geq 1$. But $|w'|$ is odd and hence $f(\mathcal{A}_{w'}) = 0$ contradicting the assumption that the formula $\phi(\overline{R}, \overline{x})$ defined the function f. ◀

Last, we want to give another inclusion result between one of our classes and a class from the Saluja et al. hierarchy.

▶ **Lemma 24.** *There exists a function F which is in $\#\Sigma_1^{\text{rel}}$ but not in $\#\Sigma_1$.*

Proof. We prove that #3DNF is not in $\#\Sigma_1$ though it belongs to $\#\Sigma_1^{\text{rel}}$. As in Example 5, we use the vocabulary $\sigma_{\#3\text{DNF}} = (D_0, D_1, D_2, D_3)$ and consider the vocabulary σ extending $\sigma_{\#3\text{DNF}}$ with built-in linear order $\leq$, BIT and min. Suppose #3DNF is definable by a σ-formula $\Phi(\overline{g}, \overline{x}) \in \Sigma_1$. To a given DNF formula, φ, with $n \geq 2$ variables, one associates a σ-structure $\mathcal{A}_\varphi$ such that the number m of satisfying assignments of φ is equal to

$$m = |\{(\overline{g}_0, \overline{a}) \mid \mathcal{A}_\varphi \models \Phi(\overline{g}_0, \overline{a})\}|$$

Let $\{0, ..., n-1\}$ be the domain of $\mathcal{A}_\varphi$. Consider the structure $\mathcal{B}$ extending $\mathcal{A}_\varphi$ with one additional element n, correctly extending the numerical predicates. Structure $\mathcal{B}$ encodes a formula φ' whose number of satisfying assignments is obviously $2m$. Formulas in Σ_1 are stable by extension, so for any fixed $(\overline{g}_0, \overline{a})$ such that $\mathcal{A} \models \Phi(\overline{g}_0, \overline{a})$, any extension $\overline{g}'_0$ of $\overline{g}_0$ on the domain $\{0, ..., n\}$ of $\mathcal{B}$ is such that $\mathcal{B} \models \Phi(\overline{g}'_0, \overline{a})$. Each $g \in \overline{g}_0$ of arity $a \geq 1$ defined on $\{0, ..., n-1\}$ can be extended on $\{0, ..., n\}$ in at least $(n+1)^{\sum_{i=1}^{a} \binom{a}{i} n^{a-i}} \geq n+1$ ways. Hence:

$$|\{(\overline{g}'_0, \overline{a}) \mid \mathcal{B} \models \Phi(\overline{g}'_0, \overline{a})\}| \geq (n+1)m > 2m$$

contradicting the assumption that $\Phi(\overline{g}, \overline{x}) \in \Sigma_1$ defines #3DNF. ◀

7 Conclusion

In this paper we have started a descriptive complexity approach to arithmetic computations. We have introduced a new framework to define arithmetic functions by counting assignments to free function variables of first-order formulae. Compared to a similar definition of Saluja et al. where assignments to free relational variables are counted, we obtain a hierarchy with a completely different structure, different properties and different problems. The main interest in our hierarchy is that it allows the classification of arithmetic circuit classes such as #AC^0, in contrast to the one from Saluja et al.

We have only started the investigation of our framework, and many questions remain open for future research:

1. Sipser proved a depth hierarchy within the Boolean class AC^0 [14]. This hierarchy can be transferred into the context of arithmetic circuits: There is an infinite depth hierarchy within #AC^0. Does this circuit hierarchy lead to a logical hierarchy within $\#\Pi_1^{\text{sk}}$? Maybe it is possible to obtain a hierarchy defined by limiting the arity of the Skolem functions.

2. The connection between $\#\mathrm{AC}^0$ and the variable hierarchy studied in Sect.5 is not clear. We think it would be interesting to study if $\#\mathrm{AC}^0$ is fully contained in some finite level of this hierarchy.
3. One of the main goals of Saluja et al. in their paper [13] was to identify feasible subclasses of #P. They showed that $\#\Sigma_0^{\mathrm{rel}}$-functions can be computed in polynomial time, but even more interestingly, that functions from a certain higher class $\#R\Sigma_2$ allow a full polynomial-time randomized approximation scheme. Are there approximation algorithms or even schemes, maybe randomized, for some of the classes of our hierarchy?
4. The most prominent small arithmetic circuit class besides $\#\mathrm{AC}^0$ is problably the class $\#\mathrm{NC}^1$ [7]. Can it be characterized in our framework or by a natural extension of it, for example by allowing generalized quantifiers? The Boolean class NC^1 is obtained by first-order formulae with Lindström quantifiers for group word problems; i.e., we have, very informally, that $\mathrm{AC}^0 = \mathrm{FO}$ and $\mathrm{NC}^1 = \mathrm{FO} + \mathrm{GROUP}$, see [5, 16].
5. In Sect. 6, we clarified the inclusion relation between the class $\#\mathrm{AC}^0$ and all classes of the Saluja et al. hierarchy, and we gave a small number of examples for (non-)inclusion results between other classes from the two different settings. We consider it interesting to extend this systematically by studying the status of all further possible inclusions between classes from our hierarchy and classes of the Saluja et al. hierarchy.
6. We consider it interesting to study systematically the role of built-in relations. E.g., Saluja et al. define their classes using only linear order, and prove the hierarchy structure given in Theorem 4. It can be shown that by adding BIT, SUCC, min and max we obtain $\#\Pi_1^{\mathrm{rel}} = \#\mathrm{P}$. How does their hierarchy change when we generally introduce SUCC or BIT?

References

1 Manindra Agrawal, Eric Allender, and Samir Datta. On TC^0, AC^0, and arithmetic circuits. *Journal of Computer and System Sciences*, 60(2):395–421, 2000.

2 Eric Allender. The permanent requires large uniform threshold circuits. *Chicago J. Theor. Comput. Sci.*, 1999. URL: `http://cjtcs.cs.uchicago.edu/articles/1999/7/contents.html`.

3 Eric Allender. Arithmetic circuits and counting complexity classes. In *Complexity of Computations and Proofs, Quaderni di Matematica*, pages 33–72, 2004.

4 D. A. Mix Barrington and N. Immerman. Time, hardware, and uniformity. In *Proceedings 9th Structure in Complexity Theory*, pages 176–185. IEEE Computer Society Press, 1994.

5 D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

6 C. Behle and K.-J. Lange. Fo[<]-uniformity. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 183–189. IEEE Computer Society Press, 2006. `doi:10.1109/CCC.2006.20`.

7 H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.

8 Stephen A. Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192. ACM, 1972.

9 Etienne Grandjean and Frédéric Olive. Graph properties checkable in linear time in the number of vertices. *J. Comput. Syst. Sci.*, 68(3):546–597, 2004.

10 Anselm Haak and Heribert Vollmer. A model-theoretic characterization of constant-depth arithmetic circuits. *CoRR*, abs/1603.09531, 2016. URL: `http://arxiv.org/abs/1603.09531`.

11 William Hesse. Division is in uniform TC^0. In *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, pages 104–114, 2001.

12 Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

13 Sanjeev Saluja, K. V. Subrahmanyam, and Madhukar N. Thakur. Descriptive complexity of #P functions. *Journal of Computer and System Sciences*, 50(3):493–505, 1995.

14 M. Sipser. Borel sets and circuit complexity. In *Proceedings 15th Symposium on Theory of Computing*, pages 61–69. ACM Press, 1983.

15 L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

16 Heribert Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.

Extending Homotopy Type Theory with Strict Equality*

Thorsten Altenkirch[1], Paolo Capriotti[2], and Nicolai Kraus[3]

1 **University of Nottingham, School of Computer Science, Nottingham, UK**
`thorsten.altenkirch@nottingham.ac.uk`
2 **University of Nottingham, School of Computer Science, Nottingham, UK**
`paolo.capriotti@nottingham.ac.uk`
3 **University of Nottingham, School of Computer Science, Nottingham, UK**
`nicolai.kraus@nottingham.ac.uk`

Abstract

In homotopy type theory (HoTT), all constructions are necessarily stable under homotopy equivalence. This has shortcomings: for example, it is believed that it is impossible to define a type of semi-simplicial types. More generally, it is difficult and often impossible to handle towers of coherences. To address this, we propose a 2-level theory which features both strict and weak equality. This can essentially be represented as *two* type theories: an "outer" one, containing a strict equality type former, and an "inner" one, which is some version of HoTT. Our type theory is inspired by Voevodsky's suggestion of a *homotopy type system* (HTS) which currently refers to a range of ideas. A core insight of our proposal is that we do not need any form of equality reflection in order to achieve what HTS was suggested for. Instead, having unique identity proofs in the outer type theory is sufficient, and it also has the meta-theoretical advantage of not breaking decidability of type checking. The inner theory can be an easily justifiable extensions of HoTT, allowing the construction of "infinite structures" which are considered impossible in plain HoTT. Alternatively, we can set the inner theory to be exactly the current standard formulation of HoTT, in which case our system can be thought of as a type-theoretic framework for working with "schematic" definitions in HoTT. As demonstrations, we define semi-simplicial types and formalise constructions of Reedy fibrant diagrams.

1998 ACM Subject Classification F.4.1 Mathematical Logic: Lambda calculus and related systems

Keywords and phrases homotopy type theory, coherences, strict equality, homotopy type system

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.21

1 Introduction: Motivations for a 2-Level System

The identity type is probably the single concept of intensional Martin-Löf type theory (MLTT) which has created most questions, stimulated most research, caused most confusion, and enabled the largest number of different views. Written $\mathsf{Id}_A(x, y)$ for elements x, y of a type A, the identity type expresses that two elements are equal in some sense and can be substituted for each other, and the elements of this type are called equalities. However, by default, it has a somewhat strange standing. On the one hand, it is not well-behaved when it comes to describing equality of functions and equality of types. Given two functions of the same type,

* This work was supported by the Engineering and Physical Sciences Research Council (EPSRC), grant reference EP/M016994/1, and by USAF, Airforce office for scientific research, award FA9550-16-1-0029.

© Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus;
licensed under Creative Commons License CC-BY

we cannot derive the principle of (naive) function extensionality, saying that the functions are equal if they are equal at every point, from the basic axioms. We also cannot show that equivalent types can be substituted for each other, although they do behave equivalently in any given situation. On the other hand, we also cannot derive the principle of unique identity proofs (UIP): by a construction of Hofmann and Streicher [11], we can not show $\mathsf{Id}_{\mathsf{Id}(x,y)}(p,q)$ for two equalities p, q. A priori, it is unclear what it should mean to have distinguishable equalities and how one can make sense of this behaviour.

As we view it, there are two major ways to remedy the situation. Both can be seen as extensions of MLTT. The principle of function extensionality can be added in any case, but after that, we have two possibilities to extend the theory further. The first is to add UIP (or, equivalently, Streicher's K) as an axiom. Let us call the resulting type theory MLTT_K. The second possibility is to consider univalence, ensuring that type equality is what one would ideally expect. This approach is taken by Homotopy Type Theory [21], and we write HoTT for the resulting theory.

One appeal of HoTT is that equalities can be seen as paths in a space, and it is even possibly to develop a lot of homotopy theory synthetically. An important insight is that, when doing homotopy theory in type theory, every statement that we make is up to homotopy, and every construction respects (homotopical) equivalence. This means that whatever we do will be "non-evil" in the sense that it can only take the homotopy type of spaces, and homotopy equivalence classes of maps, into account, and not the concrete representations of spaces or maps. Exactly this is often considered a selling point of HoTT: one often defines constructions using representatives of homotopy classes in traditional homotopy theory and is forced to show that the constructions are well-defined, i.e. do not depend on the choice of the representative. In HoTT, everything we do is automatically well-defined as we are simply not able to talk about strict properties internally.

Going back to the homotopical point of view, it is not hard to imagine that the blessing of having only constructions up to homotopy can turn out to be a curse: we are unable to make any strict statement. For example, we cannot form a type expressing that a given diagram commutes strictly; all we can do it stating that it commutes up to homotopy. Unfortunately, depending on the shape of the diagram, this will only be sufficient in the simplest cases. More often than not, it will be necessary to say that the different "pieces" (the equalities expressing commutativity) fit together. For example, the fact that a certain sub-diagram commutes can be part of the proof that the diagram commutes, but it may at the same time be derivable as the composition of the fact that other sub-diagrams commute. In this case, it is natural to require these different ways of getting a certain proof to be equal. It does not stop here; these new proofs can itself be required to be coherent, and so on. What happens here is not at all something that can only be observed in type theory. The first step becomes already apparent in the theory of monoidal categories in the form of "Mac Lane's Pengaton". On higher dimensions, it is exactly the same issue that is discussed as *homotopy commutativity versus homotopy coherence* by Lurie [16].

In general, homotopy coherence corresponds to infinite towers of coherence data, and it is a major open problem (and commonly believed to be unsolvable) to express such towers internally in HoTT. One way to avoid this problem is to restrict constructions to types of low truncation levels. As an examples, the category theory developed in [1] only considers 1-truncated types and [what corresponds to] ordinary categories. This is in many situations not satisfactory: we know that types are ∞-groupoids [15, 22], and similarly, the universe should be an $(\infty, 1)$-category. Unfortunately, we seem to have no way of expressing this internally in HoTT.

The crucial shortcoming of HoTT is that we are unable to perform some constructions which actually *seem* to be harmless as they only require finite amounts of coherences at every step. An example that has received considerable attention in the HoTT-community is the construction of Reedy fibrant n-truncated semi-simplicial types (simply referred to as *semi-simplicial types*). Let us start with Δ_+, the category of finite non-zero ordinals and strictly monotonous functions. Let us write $[\mathsf{n}]$ for the ordinal with $(n+1)$ elements. A type-valued diagram over Δ_+^{op} is a strict functor from Δ_+^{op} to the category of types. It would correspond to a type $X_{[\mathsf{n}]}$ (for simplicity written X_n) for every n, and face maps $d_i : X_{n+1} \to X_n$ for $0 \leq i \leq n$, as it is well-known that any map can be written as a composition of face maps. The problem is that we need the semi-simplicial identities (essentially a representation of the functor laws) to be strict, which we cannot express in type theory. The considered approach to avoid this problem is to only attempt internalising Reedy fibrant diagrams over Δ_+^{op}, essentially ensuring that the face maps are simple projections. Using the correspondence between fibrations and type families, a (Reedy fibrant) semi-simplicial type then corresponds to a type X_0 (the "points") on level 0. On level 1, we need a family $X_1 : X_0 \to X_0 \to \mathcal{U}$, where $\mathcal{U}$ is the universe of types. We think of X_1 as lines between types. Next, we need $X_2 : \Pi_{a,b,c:X_0} X_1(a,b) \to X_1(b,c) \to X_1(a,c) \to \mathcal{U}$, the type of fillers for triangles. Writing down the type of X_4 is already a bit tedious, but nevertheless straightforward: X_4 is a family which gives a type for any collection of four points, six lines and four triangles that form an empty tetrahedron. The long-standing open problem of homotopy type theory is to write down the type of X_n in general (up to equivalence). Perhaps surprisingly, this does not seem to be possible. What *is* possible is to generate an expression X_n for every externally fixed numeral n, such that the expressions $X_0, X_1, X_2, \ldots$ all "fit together". When one tries to do the construction for a *variable* $n : \mathbb{N}$, it does no longer type-check. The reason is that some judgmental equalities that hold in the case of a numeral n fail to hold in the case of a variable. We can try to prove in type theory that the required equalities hold up to homotopy. However, we quickly have to realize that we then also need that these equalities are coherent, and that the coherence proofs are coherent themselves, and so on; something that no one has managed to do so far. The problem is that we cannot formulate the tower of coherences that we need to prove. Morally, the required equalities *should* hold and be fully coherent just because they are trivially satisfied for each externally fixed natural number. If we can use a system where we judgmental equalities can be shown by induction, there would thus be no problem at all; however, this would require judgmental equalities to be some sort of type.

In MLTT_K, the internal equality type can be seen as an internalised version of judgmental equality. For example, a well-known meta-theoretic statement is that any equality that is constructed in the empty context is refl; that is, if we can show an equality internally without assumptions, then this equality holds judgmentally. Not surprisingly, it is possible to construct Reedy fibrant semi-simplicial types in MLTT_K. However, we can also simply define categories and functors in the naive sense, as all coherences are satisfied automatically.

The idea of a 2-level system is to combine MLTT_K and HoTT instead of viewing them as two alternative extensions of MLTT. We can describe this in two ways:

1. Start with a type theory that has axiom K and consider a "sub-theory" of types and maps that do not talk about equalities. Inside this sub-system, we can consider a new equality type and univalent universes. If we use the equality type with K of the outer system, we cannot form types that live in the inner system; however, we can reason about the inner system.
2. We may start with HoTT and try to formulate the meta-theory (in which judgmental equality lives) as a type system. It is not necessary to capture every aspect of the meta-theory in this type system; the important part is that this outer type system has

an equality type (which we call *strict equality*) satisfying K. We then have in total three equalities: the equality in HoTT; the strict equality; and the judgmental equality (which we should now refer to as *definitional equality*). From the point of view of HoTT, the strict equality and definitional equality are identical.

Considering a type theory with two equality types is not new. Our proposal is motivated by the suggestion of a *homotopy type system* (HTS) by Voevodsky [23]. However, as far as we are aware, "HTS" mostly refers to a range of ideas so far but not to a precise theory, and there is no publication that presents or analyses HTS. The core idea of HTS is to make some judgmental equalities provable. In other words, some form of equality reflection, the characteristic concept of extensional type theory, is reintroduced, in a way that is compatible with the "standard" intensional identity type. A concrete theory that could be called "HTS" is outlined in the draft [23] which, unfortunately, presents rather involved rules that would presumably be non-trivial to justify. One original goal of *Andromeda*, a project by Bauer et al. [4], was to serve as an implementation of HTS. This is currently not the case as a fibrant fragment with a univalent universe has not yet been implemented, but future developments might go into that direction.

A key observation of the current paper is that no form of equality reflection is actually required. Our proposal instead only required unique identity proofs for the strict equality type. Thus, we can avoid all the problems that are usually connected to equality reflection, such as undecidability of type checking. In contrast, the theory that we suggest is well-behaved, very close to the standard formulation of HoTT, and has straightforward semantics. One could expect that a downside of our system might be reduced expressibility compared to a theory that features equality reflection. However, we show that we can achieve in our system what HTS was suggested for: a definition of semi-simplicial types, and other constructions. This should actually not be surprising in the light of Hofmann's result [9], which states that equality reflection is conservative over MLTT_K.

Our 2-level theory can be defined as two separate type theories with a morphism between them. This actually gives a recipe for constructing a variety of reasonable 2-level theories, and the choices that can be made affect the exact abilities of the system. We believe that our 2-level theories can be used in two ways. First, we can use the outer theory as a powerful formal language to study the inner theory. For some formulations of the 2-level theory, we get a conservativity result (by an argument of the second-named author; see the forthcoming thesis [5]). This means that the inner theory is exactly HoTT as studied in the standard textbook on homotopy type theory [21] and by many authors. In a proof assistant which supports this theory, we can then implement results that so far can only be stated meta-theoretically. To give an example, it is shown in [13] that constant functions from A to B which satisfy n coherence conditions correspond to maps $\|A\|_{-1} \to B$, provided that B is n-truncated. This can be done in HoTT only if n is an externally fixed natural number. In the 2-level system, we can formalise it by taking n to be a number in the outer theory, and show that the equivalence holds in the inner theory.

Second, we can use the construction of 2-level theories to derive extensions of HoTT that allow constructions that HoTT does not allow. For example, we can assume that the natural numbers of the outer theory are *exactly* the natural numbers of the inner theory, something that is satisfied in the simplicial set model. This gives us a univalent type theory in which various concepts including semi-simplicial types can be defined.

Contributions of the paper. Summarised, the main contributions of the paper are:

- We give (for the first time) a clean presentation of a system with two equalities.

- Our theory is simple enough to have straightforward semantics. Such semantics have not yet been described for previous proposals [23, 18].
- We demonstrate how our theory allows constructions that are thought to be impossible in standard HoTT, such as semi-simplicial types [8, 18]. A partial Agda formalisation is available.
- Schematic constructions, which could so far only be given on paper, can be formalised in our system. As an example, we perform constructions of Reedy fibrant diagrams. Further work is outlined in the conclusions.

Related work. The current paper is the write-up of our presentation [3] at TYPES'15. As briefly explained above, the main difference to Voevosky's draft [23] is that we do not consider any form of equality reflection, saving us from various difficulties.

Superficially related is the construction by Maietti [17] of a *two-level foundation for constructive mathematics*. However, their motivation and goals are very different from ours, hence their system cannot be used to reconcile strict equality with univalence.

Our work is more closely related to a recent proposal by Part and Luo [18] of Logic-enriched HoTT. In their system, our strict layer of type theory is replaced by a "logic enrichment". Their proposal is limited to the construction of semi-simplicial types (corresponding to the one that we give in Section 3). It is not explained whether this can be generalised to Reedy fibrant diagrams in the sense we present in Section 4, as they use specific properties of the Δ_+ category.

Herbelin has given a construction of semi-simplicial types along the lines of the one in Section 3 in an unspecified type theory containing a "connective" for strict equality [8].

Agda formalisation. As a proof assistant based on our 2-level theory does not (yet) exist, we cannot formalise our constructions exactly as they are presented. However, we have implemented in Agda an approximation of the construction of semi-simplicial types that is given in Section 3. It can be found on GitHub at `github.com/nicolaikraus/HoTT-Agda/tree/master/nicolai/SemiSimp`. For an explanation of the relationship between the construction given in the paper and this implementation, we refer to the last remark of Section 3.

Organisation. The structure of the paper is as follows. In Section 2, we specify our 2-level theories. Section 3 explains the construction of Reedy fibrant n-truncated semi-simplicial types in a way that could *nearly* be done in homotopy type theory, and we show how the missing gap is filled by our strict equality. Then, in Section 4, we demonstrate how our theory can be used to internalise standard constructions in a fairly straightforward way. Finally, in Section 5, we outline further work and conclude the paper.

2 The Specification of a 2-Level System

In this section, we want to specify our 2-level theory (or, to be precise, our family of 2-level theories). We give two presentations: first, the semantical approach, and second, the syntactical approach. With the first approach, we explain how the theory is constructed. It also shows which choices can be made, and how models of the 2-level theory can be constructed. The syntax that we propose afterwards is based on the semantics, but fixes a precise system.

2.1 Semantical Approach

Many models of type theory consist of a category $\mathcal{C}$, modelling the category of contexts. Starting from $\mathcal{C}$, additional structures are added to model types and terms, together with the structure that is needed to model the components of the considered theory (such as universes or dependent functions). Then, a model of a 2-level theory in our sense is given by a category of contexts $\mathcal{C}$, together with *two* structures on $\mathcal{C}$ such that the first structure (taken together with $\mathcal{C}$) models HoTT, and the second structure (taken together with $\mathcal{C}$) models MLTT_K. Finally, we need a morphism between the structures in a suitable sense, describing how any type or term in HoTT can be viewed as a type or term in MLTT_K.

We make this precise using the notion of *categories with families* [6]. Let us recall the definition:

▶ **Definition 1** (CwF [6])**.** A *category with families* (CwF) is given by:

- a category $\mathcal{C}$, equipped with a distinguished terminal object 1;
- a presheaf $\mathsf{Ty} : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$;
- a presheaf $\mathsf{Tm} : \int \mathsf{Ty} \to \mathsf{Set}^{\mathrm{op}}$;
- for all $\Gamma : \mathcal{C}$ and $A : \mathsf{Ty}(\Gamma)$, an object $(\Gamma.A, \pi_A) : \mathcal{C}/\Gamma$ representing the functor $\mathcal{C}/\Gamma \to \mathsf{Set}^{\mathrm{op}}$ defined by:

$$(\Delta, \sigma) \mapsto \mathsf{Tm}_\Delta(A[\sigma]).$$

The objects of $\mathcal{C}$ are called *contexts*. Given a context Γ, the elements of $\mathsf{Ty}(\Gamma)$ are called *types*, and given a type A, the elements of $\mathsf{Tm}_\Gamma(A)$ are called *terms*.

The context $\Gamma.A$ is called the *context extension* of Γ by the type A, and π_A is the *display map* of A.

The action of Ty and Tm on morphisms is called *substitution*.

A CwF can be regarded as a model of MLTT with only *structural rules*, i.e. rules that deal with types, terms and substitutions, but no type formers (like Π or Σ types). Type formers can be postulated separately as additional structures on a CwF. For details, we refer to [6] and [10].

To model a 2-level type theory, we need to add some extra structure to a CwF:

▶ **Definition 2.** A *2-level category with families* is a CwF $\mathcal{C}$, together with:

- a presheaf $\mathsf{Ty}^{\mathrm{f}} : \mathcal{C} \to \mathsf{Set}^{\mathrm{op}}$;
- a natural transformation $|-| : \mathsf{Ty}^{\mathrm{f}} \to \mathsf{Ty}$.

Given a 2-level CwF $\mathcal{C}$, we will denote the underlying category with family by $\mathcal{C}^{\mathrm{s}}$. There is also a second CwF structure on $\mathcal{C}$, where the types are given by Ty^{f}, and terms are defined as:

$$\mathsf{Tm}^{\mathrm{f}}_\Gamma(A) = \mathsf{Tm}_\Gamma(|A|),$$

and context extension is given simply by $\Gamma.A = \Gamma.|A|$. We will denote this second CwF by $\mathcal{C}^{\mathrm{f}}$.

The map $|-|$ determines a morphism of CwF $\mathcal{C}^{\mathrm{f}} \to \mathcal{C}^{\mathrm{s}}$, which we will also denote by $|-|$.

The theory employed in this paper will be modelled by a 2-level CwF $\mathcal{C}$ where:

- C^{s} is a model of MLTT_K;
- C^{f} is a model of HoTT;
- the morphism $|-|$ preserves Π, Σ and 1 *strictly*.

Note that, crucially, equality types, although present in both CwF structures, are *not* generally preserved. This is important, because preservation of equality would mean that axiom K holds in $\mathcal{C}^{\mathrm{f}}$, which in turn would imply that $\mathcal{C}^{\mathrm{f}}$ does not admit any univalent universes containing non-propositional types.

Other type formers besides those mentioned might or might not be preserved. We say that a 2-level CwF is *strong* if $|-|$ preserves coproducts, natural numbers, and the empty type (more generally W-types, if part of the theory).

Interestingly, most of the existing models of HoTT can be naturally extended to a 2-level CwF. Most notably, the simplicial model [12] can be regarded as a 2-level CwF, where Ty is given by arbitrary (well-ordered) morphisms, Ty^{f} is the subfunctor of Ty consisting of Kan fibrations, and $|-|$ is simply the inclusion. With this setup, C^{s} is (equivalent to) a presheaf CwF, which models type theory with equality reflection (hence, in particular, MLTT_K), and C^{f} is the same as the model defined in the paper.

One can also start with an arbitrary model $\mathcal{C}$ of HoTT, then consider the presheaf category $\widehat{\mathcal{C}}$. It is perhaps not surprising that one can equip $\widehat{\mathcal{C}}$ with a 2-level CwF structure so that $\mathcal{C}$ can be recovered inside $\widehat{\mathcal{C}}^{\mathrm{f}}$. This makes it possible to use 2-level type theory to formulate and prove statements that hold in any model of HoTT, i.e. 2-level type theory can be regarded as a meta-language for HoTT.

However, the details of this construction are somewhat involved, mainly due to the strictness requirement in Definition 2. Therefore, we will not explore that direction further in this paper and refer instead to the forthcoming thesis of the second-named author [5].

2.2 Syntactical Approach

In the syntactical approach, the clear separation of a 2-level theory into two theories becomes harder to see. We do not leave as many choices open as in the semantical approach, but rather fix a concrete theory; and the choices that we make ensure that the conservativity result of the forthcoming thesis [5] applies to the presented theory.

For a precise specification, we choose a presentation in the style of [21, Appendix A.2], which considers three forms of judgments: $\Gamma \vdash \mathrm{ctx}$; $\Gamma \vdash a : A$; and $\Gamma \vdash a \equiv a' : A$. Fortunately, we do not need to give *all* the rules, as most of them are identical to those given in [21, Appendix A.2]. Thus, in most cases, it is sufficient to state the difference in order to give both an understandable and a precise specification.

The theory that we consider has the following basic types and type formers: Π, the type former of dependent functions; Σ, the type former of dependent pairs; $+$, the coproduct type former; $\mathbf{1}$, the unit type; $\mathbf{0}$, the empty type; $\mathbb{N}$, the fibrant type of natural numbers; $=$, the equality type (in the sense of HoTT); a hierarchy $\mathcal{U}_0, \mathcal{U}_1, \ldots$ of universes. So far, we can think of these as the types and type formers of HoTT. Further, we have: $+^s$, the strict coproduct; $\mathbf{0}^s$, the strict empty pretype; $\mathbb{N}^s$, the strict pretype of natural numbers; $\overset{s}{=}$, the strict equality; and hierarchy $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$ of strict universes.

Both the hierarchy $\mathcal{U}_0, \mathcal{U}_1, \ldots$ and the hierarchy $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$ are cumulative. We think of the elements of $\mathcal{U}_i$ as *fibrant types* (or simply *types*), while the elements of $\mathcal{U}_i^s$ are *pretypes*.

Recall possibility 1 from the two ways of describing a 2-level system as outlined on page 3: we can start with a type theory with K and embed HoTT later. Thus, we first consider the type theory with the basic types $\mathbf{0}^s$, $\mathbf{1}$, $\mathbb{N}^s$, with universes $\mathcal{U}_0^s, \mathcal{U}_1^s, \ldots$, and with $+^s$, Π, and Σ. All rules correspond exactly to those of [21, Appendix A.2]. For example:

- Contexts are formed using elements of $\mathcal{U}_i^s$, i.e. if Γ is a context and $\Gamma \vdash A : \mathcal{U}_i^s$, then $\Gamma.A$ is a context.
- If $\Gamma \vdash A : \mathcal{U}_i^s$ and $\Gamma.A \vdash B : \mathcal{U}_i^s$, then we have $\Gamma \vdash \Pi_A B : \mathcal{U}_i^s$.

- If $\Gamma, x : A \vdash b : B$, then we have $\Gamma \vdash \lambda x.b : \Pi_A B$.
- All further rules of Π, and all rules of Σ, $+^s$, $\mathbf{0}^s$, $\mathbf{1}$, and $\mathbb{N}^s$ are also those given in [21, A.2.4–9]. The constructors of $+^s$ are called inl^s, inr^s, and the constructors of $\mathbb{N}^s$ are called $\mathbf{0}^s$ and succ^s. We assume all the usual judgmental rules (including the judgmental η-rule for Σ).

Further, the theory has a strict identity pretype, written $\overset{s}{=}$: For any $\Gamma \vdash A : \mathcal{U}_i^s$ and $\Gamma \vdash a_1, a_2 : A$, we have $\Gamma \vdash a_1 \overset{s}{=} a_2 : \mathcal{U}_i^s$, with the introduction rule refl^s, the eliminator J^s, and the usual computation rule. For pretypes $A, B : \mathcal{U}_i^s$, we can form the pretype of strict isomorphisms, written $A \simeq^s B$ (unlike in HoTT, it is enough to have maps in both directions such that both compositions are pointwise strictly equal to the identity). However, we do *not* assume that $\mathcal{U}_i^s$ is univalent. Instead, we add the rule K^s: for A, a_1, a_2 as before, and for $\Gamma \vdash p, q : a_1 \overset{s}{=} a_2$, we have a term $\Gamma \vdash K^s(p, q) : p \overset{s}{=} q$. We also assume that $\overset{s}{=}$ satisfies the principle of function extensionality.

Note that, so far, we have not considered $\mathcal{U}_i$, $+$, $\mathbf{0}$, $\mathbb{N}$, $=$ at all. We do this now, and their rules are more subtle. The first important rule is that any type (element of $\mathcal{U}_i$) is also a pretype (element of $\mathcal{U}_i^s$), as given by the inference rule (a) below. This means that informally we can understand $\mathcal{U}_i$ as a subtype of $\mathcal{U}_i^s$.

Now, let A and B be fibrant types, i.e. $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma.A \vdash B : \mathcal{U}_i$. Then, by (a) and by the formation rule of Π, we have $\Gamma \vdash \Pi_A B : \mathcal{U}_i^s$. However, we add the rule that, under these conditions, this conclusion can be lifted to $\Gamma \vdash \Pi_A B : \mathcal{U}_i$. In other words, Π preserves types. We add the same rule for Σ, as shown by the rule (b) below. We do *not* add the same rule for $+^s$, that is, the strict sum of two types is still only a pretype. Similarly, there is no special rule for $\overset{s}{=}$: if $\Gamma \vdash a_1, a_2 : A$, it does not matter whether A is a type or only a pretype, the expression $a_1 \overset{s}{=} a_2$ is only an element of $\mathcal{U}_i^s$, not of $\mathcal{U}_i$.

In contrast, the equality type former $=$ can only be applied to elements of fibrant types; i.e. its formation rule is given by the rule (c) below. Note that there is no strict universe $\mathcal{U}_i^s$ involved:

$$\frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_i^s} \quad (a) \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma.A \vdash B : \mathcal{U}_i}{\Gamma \vdash \Sigma_A B : \mathcal{U}_i} \quad (b) \qquad \frac{\Gamma \vdash A : \mathcal{U}_i \quad \Gamma \vdash a_1, a_2 : A}{\Gamma \vdash a_1 = a_2 : \mathcal{U}_i} \quad (c)$$

The type $a_1 = a_2$ (with the constructor refl) is a pretype by rule eq:type-is-pretype, but (usually) not the same as $a_1 \overset{s}{=} a_2$. The elimination principle of $=$ only works for families of types (not in general for pretypes). This means that the usual "path induction" principle, which allows us to construct an element of $\Pi_{a_1,a_2:A}\Pi_{p:a_1=a_2} P(a_1, a_2, p)$, can only be applied if P is a family of types, i.e. $\Gamma \vdash P : (\Sigma_{a_1,a_2:A} a_1 = a_2) \to \mathcal{U}_i$. If we restrict ourselves to types, we can do everything that we can do in HoTT. In particular, we can say what it means for a function between types to be an equivalences (using $=$). We assume that the universes $\mathcal{U}_0, \mathcal{U}_1, \ldots$ are univalent, that is, the canonical map from type of equalities $A = B$ to the type of equivalences $A \simeq B$ (defined as usual in homotopy type theory) is an equivalence itself.

Similarly, the type former $+$ only allows us to form a type $A + B$ if A and B are types (elements of some $\mathcal{U}_i$), and we can only defined a function $\Pi_{x:A+B} P(x)$ with the usual induction principle if P is a family of types.

We have the type of natural numbers $\mathbb{N} : \mathcal{U}_0$ (in any context) with the constructors $\mathbf{0}$, succ, and its induction principle can only be applied to eliminate into families of types. The same is the case for $\mathbf{0}$. This completes the syntactical characterization of our 2-level system. We will usually omit the index and simply write $\mathcal{U}^s$ or $\mathcal{U}$ instead of $\mathcal{U}_i^s$ or $\mathcal{U}_i$ in the same style as it is done in [21]. A strong 2-level theory is now simply one in which $\mathbf{0}^s$ and $\mathbf{0}$, and $+^s$ and $+$, and $\mathbb{N}^s$ and $\mathbb{N}$ coincide.

▶ Remark. If A is a ("fibrant") type with elements $a_1, a_2 : A$, then we can form both the type $a_1 = a_2$ and the pretype $a_1 \overset{s}{=} a_2$. By "strict path induction" (i.e. an application of J^s), we can easily construct a function $a_1 \overset{s}{=} a_2 \to a_1 = a_2$. Consequently, strictly equal elements of a type are also homotopy-equal. This corresponds to the fact that judgmental equality in HoTT implies equality ("refl"). We cannot construct a function in the other direction, as the path induction principle J can only be applied to eliminate into types, which $a_1 \overset{s}{=} a_2$ is not. Hence, equal elements are not necessarily strictly equal. However, if we have a type which does satisfy this "equality reflection" principle, it is easy to see that the type is a set in the sense of homotopy type theory.

3 Semi-Simplicial Types

In a 2-level theory, we can define strict categories in a reasonable sense. There are a number of choices that one can make; for example, the objects could be a fibrant type or only assumed to be a pretype. Later (see Definition 5), we will give one possible concrete definition. The important thing is that the categorical equations can be required strictly; and, if we have such a strict category $\mathcal{C}$, we can easily write down the pretype of strict functors $\mathcal{C} \to \mathcal{U}$. Unfortunately, there is no general way to get an actual fibrant type of such functors.

The case where $\mathcal{C}$ is Δ^{op} (the category of finite nonempty ordinals and increasing functions) is particularly interesting since "simplicial structures" appear frequently in homotopy theory. Having a type of functors $\Delta^{\mathrm{op}} \to \mathcal{U}$ would have many potential applications; maybe most notably, one could try to internalise a constructive version of the model of univalent foundations in simplicial sets [12]. Unfortunately, it seems unreasonable to expect that such a type can be constructed. It would be a good approximation (and potentially good enough for many constructions) if one could form a type of functors $\Delta_+^{\mathrm{op}} \to \mathcal{U}$, where Δ_+ is the category of finite nonempty ordinals and *strictly* increasing functions (a more precise definition will be given below). Trying to define such a type seems more promising, since Δ_+^{op} is an *inverse category* and, if we restrict ourselves to *Reedy fibrant* functors, we can describe them by induction (see [20]).

This gave rise to the challenge of defining Reedy fibrant n-truncated semi-simplicial types (in the community often just referred to as *semi-simplicial types*) in type theory. The challenge was first raised during the special year on Univalent Foundations at the Institute for Advanced Study (Princeton, 2012–13). As briefly sketched in the introduction, a (Reedy fibrant) 0-truncated semi-simplicial type is simply a type $X_0 : \mathcal{U}$, a 1-truncated semi-simplicial type is such an X_0 together with a family $X_1 : X_0 \to X_0 \to \mathcal{U}$, and so on. Defining n-truncated semi-simplicial types as a family $\mathsf{SST} : \mathbb{N} \to \mathcal{U}$ in homotopy type theory is a famous open problem. Many attempts (see e.g. [8, 19, 14]) have not led to a solution, and at a workshop on HoTT in Warsaw (June 29–30, 2015), a clear majority of the participants expected it to be impossible.

The hard part of the construction is to define the *matching objects* M_n, that is the "full boundary" of an n-simplex, as the corresponding component of SST_n is then just given as a family $M_n \to \mathcal{U}$. A popular attempt for defining the matching objects M_n is to define the *k-skeleton* SK_n^k of SST_n by induction on k, that is, the collection of components of SST_n up to level k. As long as k is a fixed numeral, this can be done. However, if k is a variable, some crucial judgmental equalities do not hold anymore and the construction is believed to become impossible. In our 2-level theory, we can prove strict equalities (i.e. the internalisation of judgmental equality) by induction. This allows us to complete the sketched approach of defining SST in a weak sense: we construct a family $\mathsf{SST} : \mathbb{N}^s \to \mathcal{U}$. If we assume that the

strict natural numbers ($\mathbb{N}^s$) and the fibrant ones ($\mathbb{N}$) coincide, this represents a construction of n-truncated semi-simplicial types. Without this assumption and under the conjectured conservativity result [5], it internalises the result that n-truncated semi-simplicial types can be defined for an externally fixed n.

To give the precise construction, let us first note that we have the family $\mathsf{Fin} : \mathbb{N}^s \to \mathcal{U}$ of finite types (Fin_n is the type with n elements), together with the families $<^n\colon \mathsf{Fin}_n \to \mathsf{Fin}_n \to \mathcal{U}$. Let us write $\mathsf{isIncr}_{i,j}$ for the predicate

$$\mathsf{isIncr}_{i,j} : (\mathsf{Fin}_i \to \mathsf{Fin}_j) \to \mathcal{U}$$
$$\mathsf{isIncr}_{i,j}(f) :\equiv \Pi_{x,y:\mathsf{Fin}_i}(x <^i y) \to (f(x) <^j f(y)),$$

expressing that a function is strictly monotonously increasing. Let us further write $\Delta_+(i,j)$ for the type $\Sigma\,(f : \mathsf{Fin}_i \to \mathsf{Fin}_j)\,.\,\mathsf{isIncr}_{i,j}(f)$. We then have a composition operator $\circ : \Delta_+(h,i) \to \Delta_+(i,j) \to \Delta_+(h,j)$, defined separately on each of the two components. This is a representation of strictly increasing functions such that $\circ$ is strictly associative, as observed in [14]. Unsurprisingly, this is enough to make Δ_+ a *category* in the sense that we will define later (see Definition 5).[1]

In the following, we use variable names $\vec{X}$, $\vec{x}$ instead of X, x to indicate that we have an element of a nested Σ-type, i.e. a tuple. With Δ_+ at hand, we define truncated semi-simplicial types (SST) simultaneously with skeletons (SK) and the morphism part of skeletons (written $\mathsf{SK}^{\to}$). These have the following types:

$$\begin{aligned}
&\mathsf{SST} &&: \mathbb{N}^s \to \mathcal{U} && \text{— we write } \mathsf{SST}_k \text{ instead of } \mathsf{SST}(k);\\
&\mathsf{SK} &&: \Pi_{k:\mathbb{N}^s}\ \mathsf{SST}_k \to \mathbb{N}^s \to \mathcal{U} && \text{— we write } \mathsf{SK}^n_{k,\vec{X}} \text{ instead of } \mathsf{SK}(k,\vec{X},n);\\
&\mathsf{SK}^{\to} &&: \Pi_{k:\mathbb{N}^s}\Pi_{\vec{X}:\mathsf{SST}_k}\Pi_{m,n:\mathbb{N}^s}\Pi_{f:\Delta_+(m,n)}\mathsf{SK}^n_{k,\vec{X}} \to \mathsf{SK}^m_{k,\vec{X}} &&\\
& && && \text{— we write } \mathsf{SK}^{\to}_{k,\vec{X}} \text{ instead of } \mathsf{SK}^{\to}(k,\vec{X},m,n).
\end{aligned}$$

These type families can be explained as follows:

1. SST_k is the type of $(k-1)$-truncated semi-simplicial types.
2. Assume we have a $(k-1)$-truncated semi-simplicial type $\vec{X}$, where k is smaller than another given number n.[2] $\vec{X}$ allows us to form the type $\mathsf{SK}^n_{k,\vec{X}}$. This is the type of "partial boundaries" of an $(n-1)$-truncated semi-simplicial type. Intuitively, it has n points, $\binom{n}{2}$ lines, …, and $\binom{n}{k}$ cells on level $(k-1)$.
3. We think of $\mathsf{SK}_{k,\vec{X}}$ as a "functor" from Δ_+ to $\mathcal{U}$. Its morphism component is given by $\mathsf{SK}^{\to}_{k,\vec{X}}$: for any $f : \Delta_+(m,n)$, we get a function $\mathsf{SK}^n_{k,\vec{X}} \to \mathsf{SK}^m_{k,\vec{X}}$ which simply "removes" those cells that appear in the partial boundary of an $(n-1)$-simplex, but not in the partial boundary of an $(m-1)$-simplex.

At the same time as we define SST, SK and $\mathsf{SK}^{\to}$, we prove the following strict functor law for all $k,l,m,n : \mathbb{N}^s$, $\vec{X} : \mathsf{SST}_k$, and $f : \Delta_+(l,m)$, $g : \Delta_+(m,n)$:

$$\alpha_k(\vec{X},f,g) : \mathsf{SK}^{\to}_{k,\vec{X}}g \circ \mathsf{SK}^{\to}_{k,\vec{X}}f \overset{s}{=} \mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f).$$

We define all the components by induction on k as follows. In the base case, we set $\mathsf{SST}_0 :\equiv \mathbf{1}$; $\mathsf{SK}^n_{0,\star} :\equiv \mathbf{1}$; $\mathsf{SK}^{\to}_{0,\star}f :\equiv \mathsf{id}_{\mathbf{1}}$; and $\alpha_0(\star,f,g) :\equiv \mathsf{refl}^s$. In the successor case, we

[1] Note that, for technical reasons, we include the initial object Fin_0. This explains the shift by 1: we have defined $\Delta_+(i,j) :\equiv \Sigma\,(f : \mathsf{Fin}_i \to \mathsf{Fin}_j)\,.\,\mathsf{isIncr}_{i,j}(f)$ instead of $\Sigma\,(f : \mathsf{Fin}_{i+1} \to \mathsf{Fin}_{j+1})\,.\,\mathsf{isIncr}_{i,j}(f)$.

[2] The definition works for $k \geq n$, but $k < n$ is the case that is important for the intuition.

choose

$$\begin{aligned}
\mathsf{SST}_{k+1} &:\equiv \Sigma\left(\vec{X} : \mathsf{SST}_k\right).\, \mathsf{SK}^{k+1}_{k,\vec{X}} \to \mathcal{U} \\
\mathsf{SK}^{n}_{k+1,(\vec{X},Y)} &:\equiv \Sigma\left(\vec{x} : \mathsf{SK}^{n}_{k,\vec{X}}\right).\, \Pi_{f:\Delta_+(k+1,n)} Y\left(\mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})\right) \\
\mathsf{SK}^{\to}_{k+1,(\vec{X},Y)}(f,(\vec{x},h)) &:\equiv \left(\mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})\ ,\ \lambda g.\alpha_{k*}(h(f\circ g))\right)
\end{aligned}$$

Note that, in the last line, the type of the term $h(f \circ g)$ is $Y(\mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f, \vec{x}))$. However, what we need at that point is an element of the type $Y(\mathsf{SK}^{\to}_{k,\vec{X}}(g, \mathsf{SK}^{\to}_{k,\vec{X}}(f,\vec{x})))$. This is why we transport along the proof $\alpha_k(\vec{X}, f, g)$, abbreviated to α_k, which shows that the two types are strictly equal.

We omit the term for α_{k+1} as it is not insightful to write it down explicitly. It is constructed as follows. First, we note that we need a (strict) equality between pairs; the first components are (strictly) equal by α_k. When one tries to prove that the second components are (strictly) equal, one quickly realizes that what is needed is coherence for the family of strict equalities α_k: The composition $\mathsf{SK}^{\to}_{k,\vec{X}} g \circ \mathsf{SK}^{\to}_{k,\vec{X}} f \circ \mathsf{SK}^{\to}_{k,\vec{X}} e$ can be shown to be (strictly) equal to $\mathsf{SK}^{\to}_{k,\vec{X}}(g \circ f \circ e)$ in two ways, and we need that both ways are strictly equal. Of course, this follows from the fact that we have axiom K for our strict equality. We have verified this construction in Agda (see the remark below).

▶ Remark. We and many others have attempted to formalize semi-simplicial types in homotopy type theory with exactly the outlined strategy, replacing the strict law α by the usual equality type. This works in the same way until the point where we need that α_k is coherent, which is automatic in our case. Intuitively, α_k *is* coherent, and it is easy to get trapped into thinking that this coherence can just be shown simultaneously with the other four components. However, if one does this, one notices that one needs an additional coherence level for α_{k-1}, and it continues like this. Morally, all these coherences should hold, and it is very likely that we would actually be able to prove them inductively if only we were able to write them down. Unfortunately, writing them down is a problem that is very similar to defining semi-simplicial types itself. From this point of view, what the 2-level theory gives us is the possibility to prove a certain equality and *all* its higher coherences at the same time.

▶ Remark. We have now defined the n-truncated semi-simplicial types $\mathsf{SST} : \mathbb{N}_s \to \mathcal{U}$, so we may ask whether we can define a (non-truncated) type of semi-simplicial types $\mathsf{SS} : \mathcal{U}$. If we work in the strong 2-level theory (where $\mathbb{N}_s$ and $\mathbb{N}$ coincide), we can consider the homotopy limit $\mathsf{SS} : \mathcal{U}$, defined as

$$\mathsf{SS} :\equiv \Sigma\,(f : \Pi_{n:\mathbb{N}} \mathsf{SST}_n).\, \Pi_{i:\mathbb{N}} \mathsf{fst}(f(i+1)) = f(i)$$

Then, SS is indeed a (fibrant) type that encodes (Reedy fibrant) functors $\Delta_+^{\mathrm{op}} \to \mathcal{U}$.

▶ Remark. Our Agda formalisation[3] takes place within the fibrant theory. The contribution of the strict equality is completely encapsulated in a single lemma that we postulate without a formal proof. Unfortunately, simulating our 2-level system completely in Agda, although possible in principle, would be extremely cumbersome because of the need to keep track of type fibrancy manually.

[3] `https://github.com/nicolaikraus/HoTT-Agda/blob/master/nicolai/SemiSimp/SStypes.agda`

4 Reedy Fibrant Diagrams Over Inverse Categories

In Section 3, we have defined Reedy fibrant truncated semi-simplicial types using our 2-level theory. We have stayed in the fibrant theory (HoTT) as much as we could, and only used the strict theory to prove a crucially needed coherence. In this section we want to demonstrate that the 2-level theory is even more powerful if we give up this strategy of only working in the fibrant fragment whenever possible. The point is that we can derive results about HoTT without staying inside HoTT, analogous to how one can get results that respect homotopy equivalence even when certain constructions are performed on concrete spaces that only represent homotopy types.

What we claim is that, in a proof assistant implementing a 2-level type theory, we could formalize many constructions that are presented meta-theoretically in the current literature. In the current section, we will show that Reedy fibrant diagrams $I \to \mathcal{U}$ have limits in $\mathcal{U}$ if I is a finite inverse category. This is an internalised version of results discussed by Shulman [20]. Of course it generalizes the construction in Section 3, although not "literally": the truncated semi-simplicial types that we get here will look different from those constructed above.

4.1 Essentially Fibrant Pretypes and Strict Fibrations

As a preparation for our "more abstract" sample applications of the 2-level theory, we remark that it is often not necessary to know that a pretype $A : \mathcal{U}^s$ is a fibrant type. Instead, it is usually sufficient to have a fibrant type $B : \mathcal{U}$ and a strict isomorphism $A \simeq^s B$. If this is the case, we say that A is *essentially fibrant*. Clearly, every fibrant type is also an essentially fibrant pretype.

In Section 3, we have made heavy use of the fibrant finite types Fin_n (for $n : \mathbb{N}$). In a strong 2-level theory, this type coincides with the strict pretype $\mathsf{Fin}^s_n : \mathcal{U}^s$ (for $n : \mathbb{N}^s$), but this is not in general the case. We say that some pretype I is *essentially finite* if we have a number $n : \mathbb{N}^s$ and a strict isomorphism $I \simeq^s \mathsf{Fin}^s_n$.

▶ **Lemma 3.** *Let I be essentially finite and $X : I \to \mathcal{U}$ be a family of fibrant types. Then, $\Pi_{i:I} X(i)$ is essentially fibrant.*

Proof. Essential finiteness gives us a cardinality n on which we do induction. If n is $\mathbf{0}^s$, then $\Pi_{i:I} X(i)$ is strictly isomorphic to the unit type. Otherwise, we have an essentially finite I' such that $f : \mathbf{1} +^s I' \simeq^s I$, and $\Pi_{i:I} X(i)$ is strictly isomorphic to $X(f(\mathsf{inl}\ \star)) \times \Pi_{i:I'} X(f(\mathsf{inr}\ i))$, which is essentially finite by the induction hypothesis. ◀

Similar to essential fibrancy, we have the following definition:

▶ **Definition 4** (strict fibration)**.** Let $p : E \to B$ be a function (with $E, B : \mathcal{U}^s$). We say that p is a *strict fibration* if we have a family $F : B \to \mathcal{U}$ such that the fibre of p over any $b : B$ is strictly isomorphic to $F(b)$, that is, $\Pi_{b:B} \left(F(b) \simeq^s \Sigma\,(e : E)\,.\,p(e) \stackrel{s}{=} b \right)$.

From now on, we will drop the attribute *strict* and simply talk about *fibrations*. Any fibrant type family $F : B \to \mathcal{U}$ gives rise to a fibration $p : E \to B$, as it is easy to see that the first projection $(\Sigma_B F) \to B$ satisfies the given condition. Indeed, any strict fibration is isomorphic over B to a strict fibration of this form. This often allows us to assume that a given fibration has the form of a projection.

4.2 Strict Categories

We define categories in much the same way as the precategories are defined in [21], except that we use strict equality to express the laws. Since strict equality does not suffer from coherence issues, this notion of category is well-behaved. It can be applied to structures which do not have fibrant types of objects or morphisms.

▶ **Definition 5** (strict category)**.** A *strict category* $\mathcal{C}$ is given by: a pretype $|\mathcal{C}| : \mathcal{U}^s$ of *objects*; for all pairs $x, y : |\mathcal{C}|$, a pretype $C(x, y) : \mathcal{U}^s$ of *arrows* or *morphisms*; an *identity* arrow $\mathsf{id} : \mathcal{C}(x, x)$ for every object x; and a *composition* function $\circ : \mathcal{C}(y, z) \to \mathcal{C}(x, y) \to \mathcal{C}(x, z)$ for all objects x, y, z. The usual categorical laws are required to hold strictly, that is, we have strict equalities $f \circ \mathsf{id} \stackrel{s}{=} f$ and $\mathsf{id} \circ f \stackrel{s}{=} f$, as well as $h \circ (g \circ f) \stackrel{s}{=} (h \circ g) \circ f)$.

We say that a category is *essentially finite* if the pretype of objects $|\mathcal{C}|$ is essentially finite (no condition is put on the arrows).

The usual theory of categories can be reproduced in the context of strict categories. We leave it to the reader to define appropriate notions of *functor*, *natural transformation*, *limits*, *adjunctions*, and so on.

From now on, we will refer to strict categories simply as *categories*. If $\mathcal{C}$ is a category, we will often abuse notation and use $\mathcal{C}$ itself to denote its type of objects.

Another important notion is the following:

▶ **Definition 6** (reduced coslice)**.** Given a category $\mathcal{C}$ and an object $x : \mathcal{C}$, the *reduced coslice* $x /\!\!/ \mathcal{C}$ is the full subcategory of non-identity arrows in the coslice category $x/\mathcal{C}$. A concrete definition is the following. The objects of $x /\!\!/ \mathcal{C}$ are triples of an $y : |\mathcal{C}|$, a morphism $f : \mathcal{C}(x, y)$, and a proof $\neg \left(p_*(f) \stackrel{s}{=} \mathsf{id}\right)$, for all $p : x \stackrel{s}{=} y$, where p_* denotes the transport function $\mathcal{C}(x, y) \to \mathcal{C}(y, y)$. Morphisms between (y, f, s) and (y', f', s') are elements $h : \mathcal{C}(y, y')$ such that $h \circ f \stackrel{s}{=} f'$ in $\mathcal{C}$.

Note that we have a "forgetful functor" $\mathsf{forget} : x /\!\!/ \mathcal{C} \to \mathcal{C}$, given by the first projection on objects as well as on morphisms.

4.3 Inverse Categories

Classically, *inverse categories* are categories which do not contain an infinite sequence of nonidentity arrows (see [20]). We restrict ourselves to those which have *height* at most ω, and where a *rank function* is given explicitly. First, consider the category $\mathbb{N}^{s\mathrm{op}}$ which has $n : \mathbb{N}^s$ as objects, and $\mathbb{N}^{s\mathrm{op}}(n, m) :\equiv n >^s m$ (the function $>^s : \mathbb{N}^s \to \mathbb{N}^s \to \mathcal{U}^s$ is defined in the canonical way). Then, we define:

▶ **Definition 7** (inverse category)**.** We say that a category $\mathcal{C}$ is an *inverse category* if there is a functor $\varphi : \mathcal{C} \to \mathbb{N}^{s\mathrm{op}}$ which reflects identities; i.e. if we have $f : \mathcal{C}(x, y)$ and $\varphi_x \stackrel{s}{=} \varphi_y$, then we also have $p : x \stackrel{s}{=} y$ and $p_*(f) \stackrel{s}{=} \mathsf{id}$.

4.4 Reedy Fibrant Limits

Much of what is known about the category of sets in classical category theory can be extended to the category of pretypes in a given universe. For example, the following result translates rather directly:

▶ **Lemma 8.** *The universe $\mathcal{U}^s$, viewed as a category in the canonical sense, has all small limits.*

Proof. Let $\mathcal{C}$ be a category with $|\mathcal{C}| : \mathcal{U}^s$ and $\mathcal{C}(x,y) : \mathcal{U}^s$ (for all x, y). Let $X : \mathcal{C} \to \mathcal{U}^s$ be a functor. We define L to be the pretype of natural transformations $\mathbf{1} \to X$, where $\mathbf{1} : \mathcal{C} \to \mathbf{Type}$ is the constant functor on $\mathbf{1}$. Clearly, $L : \mathcal{U}^s$, and a routine verification shows that L satisfies the universal property of the limit of X. ◀

Unfortunately, the category $\mathcal{U}$ of fibrant types is not as well behaved. Even pullbacks of fibrant types are not fibrant in general (but see Lemma 9). If we have a functor $X : \mathcal{C} \to \mathcal{U}$, we can always regard it as a functor $X : \mathcal{C} \to \mathcal{U}^s$, where it does have a limit. If this limit happens to be essentially fibrant, we say that X has a *fibrant limit*. Clearly, this limit will then be a limit of the original diagram $C \to \mathcal{U}$ (note that $\mathcal{U}$ is a full subcategory of $\mathcal{U}^s$).

▶ **Lemma 9.** *The pullback of a fibration $E \to B$ along any function $f : A \to B$ is a fibration.*

Proof. We can assume that E is of the form $\Sigma\,(b : B)\,.\,C(b)$ and p is the first projection. Clearly, the first projection of $\Sigma\,(a : A)\,.\,C(f(a))$ satisfies the universal property of the pullback. ◀

Lemma 9 makes it possible to construct fibrant limits of certain "well-behaved" functors from inverse categories. The so-called *matching objects* play an important role.

▶ **Definition 10** (matching object; see [20, Chp. 11])**.** Let $\mathcal{C}$ be an inverse category, and $X : \mathcal{C} \to \mathcal{U}$ a functor. For any $z : \mathcal{C}$, we define the *matching object* M_z^X to be the (not necessarily fibrant) limit of the composition $z /\!\!/ \mathcal{C} \xrightarrow{\mathsf{forget}} \mathcal{C} \xrightarrow{X} \mathcal{U} \subset \mathcal{U}^s$.

▶ **Definition 11** (Reedy fibrant diagram; see [20, Def. 11.3])**.** Let $\mathcal{C}$ be an inverse category and $X : \mathcal{C} \to \mathcal{U}$ be a functor. We say that X is *Reedy fibrant* if, for all $z : \mathcal{C}$, the canonical map $X_z \to M_z^X$ is a fibration.

Using this definition, we can make precise the claim that we can construct fibrant limits of certain well-behaved diagrams:

▶ **Theorem 12** (see [20, Lemma 11.8])**.** *Let $\mathcal{C}$ be an essentially finite inverse category. Then, every Reedy fibrant $X : \mathcal{C} \to \mathcal{U}$ has a fibrant limit.*

Proof. By induction on the cardinality of $\mathcal{C}$. In the zero case, the limit is the unit type.

Otherwise, let us consider the rank functor $\varphi : \mathcal{C} \to \mathbb{N}^{s^{\mathrm{op}}}$. We choose an object $z : \mathcal{C}$ such that φ_z is maximal; this is possible (constructively) since $\mathcal{C}$ is assumed to be essentially finite. Let us call $\mathcal{C}'$ the category that we get if we remove z from $\mathcal{C}$; that is, we set $|\mathcal{C}'| :\equiv \Sigma\,(x : |\mathcal{C}|)\,.\,\neg(x \overset{s}{=} z)$. Clearly, $\mathcal{C}'$ is still essentially finite and inverse.

Let $X : \mathcal{C} \to \mathcal{U}$ be Reedy fibrant. We can write down the limit of X explicitly as

$$\Sigma\left(c : \Pi_{y:|\mathcal{C}|} X_y\right)\,.\,\Pi_{y,y':|\mathcal{C}|}\Pi_{f:\mathcal{C}(y,y')} X f(c_y) \overset{s}{=} c_{y'}. \tag{1}$$

Using that z has no incoming non-identity arrows, this pretype is strictly isomorphic to

$$\begin{aligned}&\Sigma\,(c_z : X_z)\,.\,\Sigma\left(c : \Pi_{y:|\mathcal{C}'|} X_y\right)\,.\\ &\qquad\left(\Pi_{y:|\mathcal{C}'|}\Pi_{f:\mathcal{C}(z,y)} X f(c_z) \overset{s}{=} c_y\right) \times \left(\Pi_{y,y':|\mathcal{C}'|}\Pi_{f:\mathcal{C}(y,y')} X f(c_y) \overset{s}{=} c_{y'}\right).\end{aligned} \tag{2}$$

Let us write L for the limit of X restricted to $\mathcal{C}'$, and let us further write p for the canonical map $p : L \to M_z^X$. Further, we write q for the map $X_z \to M_z^X$. Then, (2) is strictly isomorphic to

$$\Sigma\,(c_z : X_z)\,.\,\Sigma\,(d : L)\,.\,p(d) \overset{s}{=} q(c_z). \tag{3}$$

This is the pullback of the span $L \xrightarrow{p} M_z^X \xleftarrow{q} X_z$. By Reedy fibrancy of X, the map q is a fibration. Thus, by Lemma 9, the map from (3) to L is a fibration.

By the induction hypothesis, L is essentially fibrant. This implies that (3) is essentially fibrant, as it is the domain of a fibration whose codomain is essentially fibrant. ◀

If $\mathcal{C}$ is an inverse category, we will denote by $\mathcal{C}^{<n}$ the full subcategory of $\mathcal{C}$ consisting of all those objects of rank less than n. Correspondingly, for a given diagram X over $\mathcal{C}$, we will denote by $X|n$ the restriction of X to $\mathcal{C}^{<n}$.

4.5 Fibrant Limits and Semi-Simplicial Types

If X is a Reedy fibrant diagram over $\mathcal{C} :\equiv \left(\Delta_+^{\mathrm{op}}\right)^{<n}$, we can restrict X to $n /\!\!/ \mathcal{C}$, then take the limit of the corresponding functor. With a slight abuse of notation, we will denote such limit by M_n^X, even though X is not defined at n.

Note that a diagram X over $\left(\Delta_+^{\mathrm{op}}\right)^{<n+1}$ is Reedy fibrant if and only if its restriction to $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$ is Reedy fibrant and the map $X_n \to M_n^X$ is a fibration. Hence, to give a Reedy fibrant diagram over $\left(\Delta_+^{\mathrm{op}}\right)^{<n+1}$ is the same as to give a Reedy fibrant diagram X over $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$, together with a fibration Y over M_n^X. We will refer to this extended diagram as $\langle X, Y\rangle$. By mutual induction on the natural number n, we can define a type SST_n, and a function SK_n from SST_n to diagrams over $\left(\Delta_+^{\mathrm{op}}\right)^{<n}$. We start with with $\mathsf{SST}_0 :\equiv \mathbf{1}$ and $\mathsf{SK}_0(\star)$ set to the trivial diagram over $\left(\Delta_+^{\mathrm{op}}\right)^{<0}$. Then, we set

$$\mathsf{SST}_{n+1} :\equiv \Sigma\,(X : \mathsf{SST}_n)\,.\,(M_n^{\mathsf{SK}_n X} \to \mathcal{U}) \qquad \text{and} \qquad \mathsf{SK}_{n+1}(X, Y) :\equiv \langle X, Y\rangle.$$

Above, we write M_n^A to mean the type given by Theorem 12 which is strictly isomorphic to the matching object of A at n (which would otherwise only be a pretype).

This gives us a succinct alternative to the construction of Section 3, where most of the hard work is encapsulated in the use of Theorem 12.

5 Conclusions and Further Work

In the previous two sections, we have demonstrated how our 2-level theories can be used in two ways. First, our framework offers reasonable, easily justifiable ways of extending homotopy type theory. Second, we can internalise results about homotopy type theory that, before, could only be stated meta-theoretically. In a suitable proof assistant which implements a 2-level theory, we could formalize many constructions that can at the moment only be done on paper. Our current article offers a demonstration of this possibility: we have shown how some of the constructions about fibrant limits and diagrams can be internalised. From here, we could go into several directions. We could, for example, internalise Shulman's result that diagrams over a model of type theory form again a model, preserving univalence [20]. Of course, for such an internalisation, we need to be careful to formulate all definitions and results constructively.

A more modest but (as we believe) worthwhile next goal is the construction of fibrant replacements. With this, we can internalise the proof that any type carries the structure of an ∞-groupoid (a Kan semi-simplicial type), as it is given in [13, Remark and Corollary 16]. To do this, we would first define an ∞-groupoid to be a Reedy fibrant semi-simplicial type $X : \Delta_+^{\mathrm{op}} \to \mathcal{U}$ such that every fibration from X_n to a horn is an equivalence (in the sense of homotopy type theory). We can then, for a type $A : \mathcal{U}$, consider the semi-simplicial type Eq_A, defined to be the Reedy fibrant replacement of the functor that is constantly A.

It is shown in [13] that Eq_A is an ∞-groupoid in our sense, and the argument can easily be internalised. This construction is in fact not difficult and has in the current paper been omitted solely for reasons of space.

Our next significant project, supported by the 2-level theory, is the development of $(\infty, 1)$-category theory. By an $(\infty, 1)$-category, which could also be called a *Segal type*, we mean a Reedy fibrant semi-simplicial type X for which the usual "Segal maps" $X_n \to X_1 \times_{X_0} \ldots \times_{X_0} X_1$ are equivalences. It is likely that it is necessary to add degeneracies, and we expect that this can be done in the way presented by Harpaz [7].

We believe that it is important to develop a theory of $(\infty, 1)$-categories type-theoretically, because the universe itself should be an $(\infty, 1)$-category; we expect that many infinite coherence problems become approachable if we can set up some basic infrastructure, so that towers of coherences could be formulated and handled in a clean way.

The most important application that we currently have in mind is the specification of *higher inductive types* (HITs). Although HITs are used frequently in the literature on homotopy type theory, we do not have a general syntactical specification yet. The approach to define a general syntactical framework of HITs that is used in [2] seems to be promising, but suffers from the issue that an unmanageable number of coherences needs to be handled manually. We expect and hope that this can be resolved with the framework of $(\infty, 1)$-categories that we plan to develop.

References

1. Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science (MSCS)*, pages 1–30, Jan 2015. doi:10.1017/S0960129514000486.
2. Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, and Fredrik Nordvall Forsberg. Towards a theory of higher inductive types. Presentation at TYPES'15, Tallinn, Estonia, 20 May 2015.
3. Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Infinite structures in type theory: Problems and approaches. Presented at TYPES'15, Tallinn, Estonia, 20 May 2015.
4. Andrej Bauer, Gaëtan Gilbert, Philipp Haselwarter, Matija Pretnar, and Chris Stone. Andromeda. Implementation of a type theory with equality reflection, ongoing project. URL: http://andromedans.github.io/andromeda/.
5. Paolo Capriotti. *Models of Type Theory with Strict Equality*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2016. In preparation.
6. Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs (TYPES)*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer-Verlag, 1995. doi:10.1007/3-540-61780-9_66.
7. Yonatan Harpaz. Quasi-unital ∞–categories. *Algebraic & Geometric Topology*, 15(4):2303–2381, 2015.
8. Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science (MSCS)*, pages 1–16, Mar 2015. doi:10.1017/S0960129514000528.
9. Martin Hofmann. Conservativity of equality reflection over intensional type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs (TYPES)*, volume 1158 of *Lecture Notes in Computer Science*, pages 153–164. Springer-Verlag, 1995. doi:10.1007/3-540-61780-9_68.
10. Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

11 Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Venice Festschrift*, pages 83–111. Oxford University Press, 1996.

12 Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations. *ArXiv e-prints*, Nov 2012.

13 Nicolai Kraus. The general universal property of the propositional truncation. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *Types for Proofs and Programs (TYPES)*, volume 39 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111–145, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.TYPES.2014.111.

14 Nicolai Kraus. *Truncation Levels in Homotopy Type Theory*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2015.

15 Peter LeFanu Lumsdaine. *Higher Categories from Type Theories*. PhD thesis, Carnegie Mellon University, 2010.

16 Jacob Lurie. *Higher Topos Theory*. Annals of Mathematics Studies. Princeton University Press, Princeton, 2009.

17 Maria Emilia Maietti. A minimalist two-level foundation for constructive mathematics. *Annals of Pure and Applied Logic*, 160(3):319–354, 2009. Computation and Logic in the Real World: CiE 2007. doi:10.1016/j.apal.2009.01.006.

18 Fedor Part and Zhaohui Luo. Semi-simplicial types in logic-enriched homotopy type theory. *CoRR*, abs/1506.04998, 2015. URL: http://arxiv.org/abs/1506.04998.

19 Michael Shulman. Homotopy type theory should eat itself (but so far, it's too big to swallow). Blog post, homotopytypetheory.org/2014/03/03/hott-should-eat-itself, 3 Mar 2014.

20 Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, pages 1–75, Jan 2015. doi:10.1017/S0960129514000565.

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

22 Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011. doi:10.1112/plms/pdq026.

23 Vladimir Voevodsky. A simple type system with two identity types, 2013. Unpublished note.

The Seifert–van Kampen Theorem in Homotopy Type Theory*

Kuen-Bang Hou (Favonia)[†1] **and Michael Shulman**[‡2]

1 **Department of Computer Science, Carnegie Mellon University, Pittsburgh, USA**
favonia@cs.cmu.edu

2 **Department of Mathematics and Computer Science, University of San Diego, USA**
shulman@sandiego.edu

Abstract

Homotopy type theory is a recent research area connecting type theory with homotopy theory by interpreting types as spaces. In particular, one can prove and mechanize type-theoretic analogues of homotopy-theoretic theorems, yielding "synthetic homotopy theory". Here we consider the Seifert–van Kampen theorem, which characterizes the loop structure of spaces obtained by gluing. This is useful in homotopy theory because many spaces are constructed by gluing, and the loop structure helps distinguish distinct spaces. The synthetic proof showcases many new characteristics of synthetic homotopy theory, such as the "encode-decode" method, enforced homotopy-invariance, and lack of underlying sets.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases homotopy type theory, fundamental group, homotopy pushout, mechanized reasoning

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.22

1 Introduction

Homotopy type theory is an emerging research area which draws ideas from type theory, homotopy theory and category theory [18, 20, 2, 19, 10, 8, 7]. Most of the current research has focused on an extension of Martin-Löf type theory by Voevodsky's *univalence axiom* and *higher inductive types* [18], interpreting types as "spaces up to homotopy", and Martin-Löf's identification type as a space of "homotopical paths". One of the more intriguing applications of this theory is *synthetic homotopy theory*: proving and mechanizing type-theoretic versions of theorems in classical homotopy theory [18, 15, 12, 14, 13, 9]. Upon translation through the homotopy-theoretic semantics of type theory [10], these yield proofs of the corresponding classical results, sometimes involving significant new insights [17].

* We thank Ulrik Buchholtz, Floris van Doorn, Daniel R. Licata, and anonymous reviewers for their useful feedback on earlier drafts.

† Sponsored by the National Science Foundation under grant numbers CCF-1116703, W911-NF0910273 and W911-NF1310154; and Air Force Office of Scientific Research under grant numbers FA-95501210370 and FA-95501510053. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

‡ Supported by the National Science Foundation under a postdoctoral fellowship and agreement No. DMS-1128155. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

© Kuen-Bang Hou (Favonia) and Michael Shulman;
licensed under Creative Commons License CC-BY

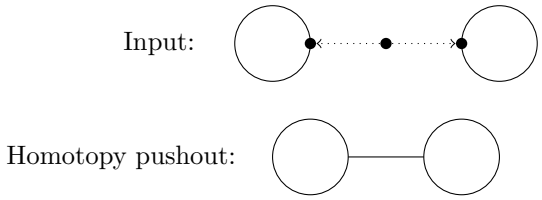


Figure 1 A homotopy pushout.

In this paper we advance the program of synthetic homotopy theory by proving and mechanizing the *Seifert–van Kampen theorem*, which computes the fundamental group of a homotopy pushout. The *fundamental group* $\pi_1(X)$ is a homotopy-theoretic invariant of a space X that measures intuitively "how many loops" it contains. For instance, the fundamental group of a circle is the integers $\mathbb{Z}$, because one can loop around the circle any number of times (in either direction); while the fundamental group of a torus (the surface of a donut) is $\mathbb{Z} \times \mathbb{Z}$, because one can either loop around the outer edge or through the hole (any number of times each).

A *homotopy pushout* is a way of gluing two spaces together to produce a new one, by specifying an inclusion of a common subspace that should be glued together. For instance, given two circles and a specified point on each, we can glue the two points together with a "bridge", producing a "barbell" shape; see Figure 1. Many of the spaces of interest to homotopy theory can be obtained by gluing together intervals, discs, and higher-dimensional discs (such gluings are called *cell complexes*); thus it is obviously of interest to calculate invariants of such gluings, such as the fundamental group.

The Seifert–van Kampen theorem tells us how to compute the fundamental group of a homotopy pushout; that is, it tells us how many loops there are in a glued space. In the example of Figure 1, a loop in the homotopy pushout can go around one circle any number of times (in either direction), then around the other circle any number of times, then back around the *first* circle some other number of times, and so on. More precisely, the fundamental group of the figure-eight or barbell is the *free product* of the fundamental groups of the two circles ($\mathbb{Z}$ and $\mathbb{Z}$), which is the coproduct in the category of groups. More generally, the Seifert–van Kampen theorem says that if we glue two spaces B and C together along a connected space A,[1] then the fundamental group $\pi_1(B \sqcup^A C)$ is the *amalgamated free product* (the pushout in the category of groups) of $\pi_1(B)$ and $\pi_1(C)$ over $\pi_1(A)$.

In fact, the full Seifert–van Kampen theorem applies also in the case when A is not connected. This requires replacing fundamental groups $\pi_1(X)$ by fundamental *groupoids* $\Pi_1 X$, which keep track of loops at different basepoints. If a space is not connected, such as the disjoint union of a circle with a point, then different numbers of loops may be possible depending on where we start. The fundamental groupoid actually records all *paths* between points, thus including loops starting at all points and also the information about which pairs of points are connected. The full SvKT says that the functor Π_1 takes homotopy pushouts to pushouts of groupoids.

Stating and proving this theorem in homotopy type theory is a bit subtle for several reasons. According to the homotopy-theoretic interpretation of type theory, types act like ∞-groupoids, and ordinary groupoids can be identified with the "1-truncated" types. Under

[1] The theorem is traditionally stated for a topological space X which is the union of two open subspaces U and V, but in homotopy-theoretic terms this is just a convenient way of ensuring that X is the homotopy pushout of U and V over $U \cap V$.

this interpretation, the Π_1 is represented by a type constructor called the 1-truncation, which is a left adjoint to the inclusion of 1-truncated types into all types. Since left adjoints preserve pushouts, the SvKT appears to follow trivially.

However, this form of SvKT is not actually particularly useful. We were originally interested in the fundamental *group* $\pi_1(X)$, which is a hom-set in the fundamental groupoid $\Pi_1 X$. If $\Pi_1 X$ is represented by a 1-truncated type, then its "hom-sets" are its path spaces (Martin-Löf's identity types), and so to find $\pi_1(X)$ when X is a pushout, we must compute the path space of a pushout. But computing the path space of a pushout is what the SvKT was supposed to do *for* us! So this version of the theorem has really just shifted the burden of calculation elsewhere.

To obtain a more calculationally useful version of SvKT, we represent groupoids more "analytically" with a type of objects and dependent types of morphisms. For $\Pi_1 X$, the type of objects is X itself, and the hom-set between $x, y : X$ is the 0-truncation (set of connected components) of the path-space from x to y. (The connection between the two fundamental groupoids is that in the language of [18, Chapter 9], the latter $\Pi_1 X$ is a "pregroupoid" whose "Rezk completion" [1] has the 1-truncation of X as its type of objects. Informally, this means that we force the type family $\Pi_1 X$ to coincide with the identity type.)

Now our goal is to calculate this truncated path space, given an expression of X as a homotopy pushout. For this we use the "encode-decode method" [15, 18]. The idea of this method is to define a type family code indexed by pairs of elements of X, using the recursion principle of X coming from its expression as a homotopy pushout; and then show (using the analogous induction principle of X, along with path induction) that this family of "codes for paths" is equivalent to the actual family of truncated path spaces.

We will do this in Section 3, after a brief review of homotopy type theory in Section 2. However, there is one further valuable refinement. The description of code in Section 3 is not maximally explicit in all cases, because it incorporates $\pi_1(X)$ through "homotopical magic". To rectify this, in Section 4 we prove an improved version of SvKT where X is equipped with an arbitrary type of "base points". Some example applications can be found at the ends of Section 3 and Section 4.

All the theorems of the paper have been mechanized in the proof assistant Agda [16]; the code can be found at `https://github.com/HoTT/HoTT-Agda/blob/1.0/Homotopy/VanKampen.agda`. We end in Section 5 with some remarks on the mechanization and how it differs from the informal treatment.

Note that most of this paper appeared previously in [18]; the relevant section §8.7 therein was in fact written by the present authors. However, since that book was self-published and never peer-reviewed, it should be regarded as a thesis or a preliminary report rather than a publication.

2 Homotopy Type Theory

As in [18], we will work in Martin-Löf type theory extended with Voevodsky's univalence axiom and some higher inductive types. For a type A and elements $x, y : A$, we write the identification type as $x =_A y$ or just $x = y$, and often refer to its elements as *paths*. The defining feature of homotopy type theory is that $x = y$ might have more than one element.

The induction principle for $x = y$, which we refer to as Id-induction or path induction, says that if $D : \prod_{(x,y:A)}(x = y) \to \mathcal{U}$, then to define $d : \prod_{(x,y:A)} \prod_{(p:x=y)} D(x, y, p)$ it suffices to define $r : \prod_{(x:A)} D(x, x, \mathsf{refl}_x)$. From this we can construct all the operations of a higher groupoid on A; for instance, given $p : x = y$ and $q : y = z$ we have their

concatenation $p \cdot q : x = z$ (identification is transitive), and for any $p : x = y$ we have its *inverse* or *opposite* $p^{-1} : y = x$ (identification is symmetric). We also have the operation of *transport* (a.k.a. substitution): given $C : A \to \mathcal{U}$ and $u : C(x)$, for any $p : x =_A y$ we have $\mathsf{transport}^C(p, u) : C(y)$. Finally, for any $f : A \to B$ and $p : x =_A y$, we can define $\mathsf{ap}_f(p) : f(x) =_B f(y)$ (functions respect identifications).

A type A is called a *mere proposition* (or simply a *proposition*) if it has at most one element, i.e. $\prod_{x,y:A} x = y$. It is called a *set* if it satisfies Uniqueness of Identity Proofs, i.e. if $\prod_{x,y:A} \prod_{p,q:x=y} p = q$; or equivalently if each type $x = y$ is a proposition. Propositions and sets are the first two rungs on an infinite ladder of "n-types" [18, Chapter 7], and as such are also called (-1)-*types* and 0-*types* respectively.

If we have two types A and B and functions $f : A \to B$ and $g : B \to A$ such that $\prod_{a:A} g(f(a)) = a$ and $\prod_{b:B} f(g(b)) = b$, then A and B are *equivalent*, written $A \simeq B$. (This is not the definition of "equivalence" — see [18, Chapter 4] for that — but it is how we generally produce equivalences.) Since types are elements of a universe type, we also have the path type $A = B$, with a canonical map $(A = B) \to (A \simeq B)$ since identified types are equivalent; the *univalence axiom* says that this canonical map is itself an equivalence, so that equivalent types are identified.

Higher inductive types (HITs) are a generalization of inductive types that allow constructors which generate new identifications (paths) in addition to elements. For instance, the circle $\mathbb{S}^1$ is a HIT generated by a point $b : \mathbb{S}^1$ and a path $l : b = b$. Note that the path l is "new" and not identified with refl_b (at least, not *a priori* — proving that it is definitely unequal to refl_b is a significant theorem [15]).

The central HIT for us will be the *pushout* $B \sqcup^A C$ of two functions $f : A \to B$ and $g : A \to C$, which is generated by the following constructors:

- $i : B \to B \sqcup^A C$,
- $j : C \to B \sqcup^A C$, and
- for all $x : A$, a path $h(x) : i(f(x)) = j(g(x))$.

As in Fig. 1, the paths $h(x)$ form the "glue", or the "handle" of the barbell. We thus have a commutative diagram

$$\begin{array}{ccc} A & \xrightarrow{\quad g \quad} & C \\ {\scriptstyle f}\downarrow & & \vdots\,{\scriptstyle j} \\ B & \xrightarrow[\quad i \quad]{} & B \sqcup^A C \end{array}$$

that is universal, in the category-theoretic sense. This follows from the type-theoretic induction principle of $B \sqcup^A C$, which says (slightly informally) that given a family $D : B \sqcup^A C \to \mathcal{U}$, to define $d : \prod_{(p:B\sqcup^A C)} D(p)$ it suffices to define $m : \prod_{(b:B)} D(i(b))$ and $n : \prod_{(c:C)} D(j(c))$ which "agree over $h(x)$" for all $x : A$. Details can be found in [18, Chapter 6]. In particular, the "recursion principle" (the case where D is non-dependent) says that to define a map $d : B \sqcup^A C \to D$, it suffices to give maps $m : B \to D$ and $n : C \to D$ and a path $mf = ng$; this is the "existence" part of the universal property of a pushout.

Other important HITs are the *propositional truncation* and the *set-truncation*. The propositional truncation of a type A is a type $\|A\|_{-1}$ that is a proposition, together with a map $|-|_{-1} : A \to \|A\|_{-1}$ that is universal among maps from A to propositions. Informally, $\|A\|_{-1}$ is "$\mathbf{0}$ if A is empty and $\mathbf{1}$ if A is inhabited". Similarly, the set-truncation of A is a type $\|A\|_0$ that is a set, together with a map $|-|_0 : A \to \|A\|_0$ that is universal among maps from A to sets; we think of it as "the set of connected components of A". See [18, Chapter 7] for

how to construct these truncations as HITs, as well as a generalization to the n-truncation (the 1-truncation was mentioned in the introduction).

Given a function $f : A \to B$ and a point $b : B$, the *fiber* of f over b is $\mathsf{fib}_f(b) :\equiv \sum_{(a:A)} f(a) = b$. We say f is an *embedding* if $\mathsf{fib}_f(b)$ is a proposition for all $b : B$, and 0-*truncated* if $\mathsf{fib}_f(b)$ is a set for all $b : B$. Dually, we say f is *surjective* if each $\|\mathsf{fib}_f(b)\|_{-1}$ is contractible (equivalent to $\mathbf{1}$), and *connected* (or 0-connected for emphasis) if each $\|\mathsf{fib}_f(b)\|_0$ is contractible. In particular, a type A is *connected* if the unique function $A \to \mathbf{1}$ is connected, which is equivalent to saying that $\|A\|_0$ is contractible — that is, A has exactly one connected component.

The set-truncation is also how we define the fundamental group and the fundamental groupoid. Given a type A and a point $a : A$, we write $\pi_1(A, a) :\equiv \|a = a\|_0$. (Often one writes simply $\pi_1(A)$, since many types have canonical "basepoints" a. Moreover, if A is connected, we have $\|a = b\|_{-1}$ for all $a, b : A$, and hence $\|\pi_1(A, a) \simeq \pi_1(A, b)\|_{-1}$ as well; so up to "propositional equivalence" the choice of a doesn't matter.) And given just a type A, for any points $x, y : A$ we write $\Pi_1 A(x, y) :\equiv \|x = y\|_0$; this defines the "hom-sets" of a groupoid with A as its type of objects. Note that we have induced groupoid operations

$$\begin{aligned}
(- \cdot -) &: \Pi_1 X(x, y) \to \Pi_1 X(y, z) \to \Pi_1 X(x, z) \\
(-)^{-1} &: \Pi_1 X(x, y) \to \Pi_1 X(y, x) \\
\mathsf{refl}_x &: \Pi_1 X(x, x) \\
\mathsf{ap}_f &: \Pi_1 X(x, y) \to \Pi_1 Y(f(x), f(y))
\end{aligned}$$

for which we use the same notation as the corresponding operations on paths.

The set-truncation also allows us to define quotients of equivalence relations on sets. If A is a set and $R : A \to A \to \mathcal{U}$ is an equivalence relation, then its "homotopy coequalizer" is the HIT generated by

- A quotient map $q : A \to Q$, and
- For each $a, b : A$ such that $R(a, b)$, a path $q(a) = q(b)$.

In general, Q will not be a set; we define the *set-quotient* of R to be its set-truncation $\|Q\|_0$. This has the usual universal property of a quotient with respect to other sets [18, §6.10].

In fact, the definition of the set-quotient makes sense even when A is not a set and when R is not an equivalence relation. That is, for any type A and any type family $R : A \to A \to \mathcal{U}$, we can define its homotopy coequalizer and then set-truncate it; we still call the result the *set-quotient* of A by R. This can be identified with a more usual sort of quotient in the following way:

1. (-1)-truncate the types $R(x, y)$, so that the resulting type family $\|R\|_{-1}$ lands in the type of propositions;
2. Since the type of propositions is a set (by univalence), we can factor $\|R\|_{-1}$ through $\|A\|_0$ to obtain a binary relation on the set $\|A\|_0$;
3. Now consider the quotient of $\|A\|_0$ by the equivalence relation generated by this relation, i.e. its closure under reflexivity, symmetry, and transitivity.

This procedure is fairly complicated, which means on the one hand that it is convenient to have a simpler construction of the set-quotient, but on the other hand that it can be difficult to describe and calculate concretely. This will become relevant in Section 4.

2.1 Encode-decode Proof Style

Many theorems in homotopy type theory, including the Seifert–van Kampen, can be phrased as an equivalence between an abstract, general description X we wish to understand (often a

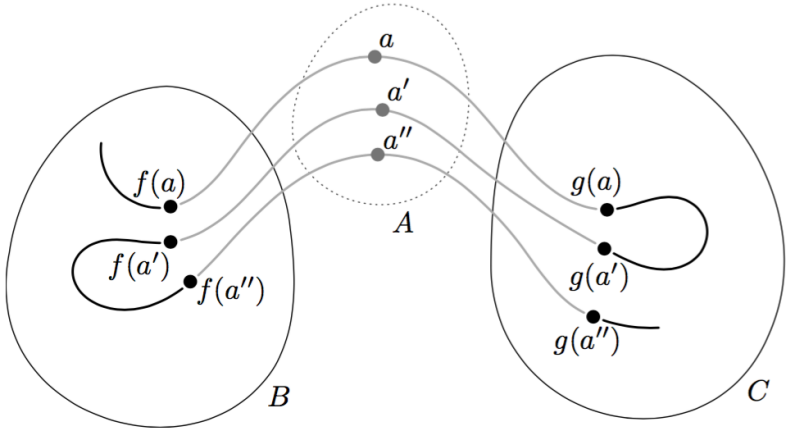


Figure 2 A zigzagging path in the pushout $P :\equiv B \sqcup^A C$.

family of path-spaces or truncated path-spaces), and a concrete, combinatorial description, which we call code. Recall that an equivalence, as mentioned above involves two functions with the proof of their mutual invertibility. We call the function from X to code "encode", and the other "decode". The encode-decode proof style essentially fills in the components of an equivalence one by one:

1. Define the code that will be equivalent to the X we wish to understand. If X is a family of (truncated) path-spaces in some higher inductive type P, then code is usually a type family over P defined using the recursion principle of P.
2. Define a encode function from X to code, and a decode function from code to X. Generally the encode function is immediate from path induction, while decode requires a further induction over the base space P.
3. Show encode and decode are inverse to each other. Again, one direction of this is usually easy, while the other requires an induction.

For the rest of the paper we will follow this recipe.

3 Naive Seifert–van Kampen Theorem

Let $f : A \to B$ and $g : A \to C$ be given functions, and let $P :\equiv B \sqcup^A C$ be their pushout. In Section 3.1 we will define the family $\mathsf{code} : P \to P \to \mathcal{U}$, and in Section 3.2 we will prove that it characterizes the truncated path spaces. In Section 3.3 we will apply this result to calculate some fundamental groups, thereby explaining why it can be regarded as a Seifert–van Kampen theorem.

3.1 Definition of code

To explain the underlying idea of code, first note that that a path from $b : B$ to $c : C$ in the pushout P can be obtained by going first from b to $f(a)$ for some $a : A$, which according to the pushout glue can be identified with $g(a)$, then going from $g(a)$ to c. However, this is not the only way to get from b to c; we might do this to get from a b to a $c' : C$, and then go back from c' to a $b' : B$ in the opposite way (using a different $a' : A$), then back from b' to $c : C$ using a third $a'' : A$; and we could have arbitrarily many such zigzags. See Figure 2. Moreover, this even gives us new ways to get from a point $b : B$ to another point $b' : B$, by

going across to C and back any number of times. And when comparing two such zigzags, we have to compare paths in B and C, but we might also relate the intermediate points in A.

To see how this is related to the classical Seifert–van Kampen theorem, consider the case of paths from $f(a_0)$ to $f(a_0)$ for some fixed $a_0 : A$. These will consist of zigzags of paths passing through B and C, connected to each other through A. If A is connected (as it usually is in classical algebraic topology), then the other points $a : A$ occurring in these zigzags can be identified with a_0 along some path, and so the paths in B and C reduce essentially to loops at $f(a_0)$ and $g(a_0)$ respectively. Thus, such a zigzag is a "word" obtained by "freely concatenating" loops in B and C; but it turns out that loops in A at a_0 can be shifted back and forth. This describes essentially the "amalgamated free product" $\pi_1(B) *_{\pi_1(A)} \pi_1(C)$ appearing in the classical Seifert–van Kampen theorem. However, in order to prove this theorem using the encode-decode method, we need to generalize it to characterize *all* the paths, not just the loops at the base points.

Formally, we define the combinatorial description $\mathsf{code} : P \to P \to \mathcal{U}$ by double recursion on P. In other words, we first apply the recursion principle to the first P in the type of code, concluding that it suffices to define maps $\mathsf{code}_B : B \to P \to \mathcal{U}$ and $\mathsf{code}_C : C \to P \to \mathcal{U}$ that agree on A. Then we apply the recursion principle again for each $b : B$ and each $c : C$, so that to define code_B it suffices to define maps $B \to B \to \mathcal{U}$ and $B \to C \to \mathcal{U}$ that agree in $B \to A \to \mathcal{U}$, and similarly for code_C. The definition of these maps is where we actually put in the zigzags. Finally, we apply the induction principle to each $a : A$ to determine what it means for code_B and code_C to agree on A. When this is all reduced out using the theorems of [18, Chapter 2] (which use function extensionality and the univalence axiom), it suffices for us to give the following.[2]

- $\mathsf{code}(i(b), i(b'))$ is a set-quotient of the type of sequences

  $$(b, p_0, x_1, q_1, y_1, p_1, x_2, q_2, y_2, p_2, \dots, y_n, p_n, b')$$

  where
  - $n : \mathbb{N}$
  - $x_k : A$ and $y_k : A$ for $0 < k \leq n$
  - $p_0 : \Pi_1 B(b, f(x_1))$ and $p_n : \Pi_1 B(f(y_n), b')$ for $n > 0$, and $p_0 : \Pi_1 B(b, b')$ for $n = 0$
  - $p_k : \Pi_1 B(f(y_k), f(x_{k+1}))$ for $1 \leq k < n$
  - $q_k : \Pi_1 C(g(x_k), g(y_k))$ for $1 \leq k \leq n$

  The quotient is generated by the following identifications:

  $$(\dots, q_k, y_k, \mathsf{refl}_{f(y_k)}, y_k, q_{k+1}, \dots) = (\dots, q_k \centerdot q_{k+1}, \dots)$$
  $$(\dots, p_k, x_k, \mathsf{refl}_{g(x_k)}, x_k, p_{k+1}, \dots) = (\dots, p_k \centerdot p_{k+1}, \dots)$$

  (see Remark 1 below). Note that the type of such sequences is a little subtle to define precisely, since the types of p_k and q_k depend on x_k and y_k; the reader may undertake it as an exercise, or refer to the Agda mechanization.
- $\mathsf{code}(j(c), j(c'))$ is identical, with the roles of B and C reversed. We likewise notationally reverse the roles of x and y, and of p and q.
- $\mathsf{code}(i(b), j(c))$ and $\mathsf{code}(j(c), i(b))$ are similar, with the parity changed so that they start in one type and end in the other.

[2] Since code is a "curried" function of two variables and we are using standard mathematical function application notation, we ought technically to write $\mathsf{code}(a)(b)$; but as in [18] we will instead write this as $\mathsf{code}(a, b)$.

- For $a : A$ and $b : B$, we require an equivalence

$$\mathsf{code}(i(b), i(f(a))) \simeq \mathsf{code}(i(b), j(g(a))). \tag{1}$$

  We define this to consist of the two functions defined on sequences by

$$(\dots, y_n, p_n, f(a)) \mapsto (\dots, y_n, p_n, a, \mathsf{refl}_{g(a)}, g(a)),$$
$$(\dots, x_n, p_n, a, \mathsf{refl}_{f(a)}, f(a)) \leftarrow\!\shortmid (\dots, x_n, p_n, g(a)).$$

  Both of these functions are easily seen to respect the equivalence relations, and hence to define functions on the types of codes. The left-to-right-to-left composite is

$$(\dots, y_n, p_n, f(a)) \mapsto (\dots, y_n, p_n, a, \mathsf{refl}_{g(a)}, a, \mathsf{refl}_{f(a)}, f(a))$$

  which is equal to the identity by a generating path of the quotient. The other composite is analogous. Thus we have defined an equivalence (1).
- Similarly, we require equivalences

$$\mathsf{code}(j(c), i(f(a))) \simeq \mathsf{code}(j(c), j(g(a)))$$
$$\mathsf{code}(i(f(a)), i(b)) \simeq \mathsf{code}(j(g(a)), i(b))$$
$$\mathsf{code}(i(f(a)), j(c)) \simeq \mathsf{code}(j(g(a)), j(c))$$

  all of which are defined in exactly the same way (the second two by adding reflexivity terms on the beginning rather than the end).
- Finally, we need to know that for $a, a' : A$, the following diagram commutes:

$$\begin{array}{ccc} \mathsf{code}(i(f(a)), i(f(a'))) & \longrightarrow & \mathsf{code}(i(f(a)), j(g(a'))) \\ \downarrow & & \downarrow \\ \mathsf{code}(j(g(a)), i(f(a'))) & \longrightarrow & \mathsf{code}(j(g(a)), j(g(a'))) \end{array} \tag{2}$$

  This amounts to saying that if we add something to the beginning and then something to the end of a sequence, we might as well have done it in the other order.

▶ Remark 1. One might expect to see in the definition of code some additional generating equations for the set-quotient, such as

$$(\dots, p_{k-1} \cdot \mathsf{ap}_f(w), x'_k, q_k, \dots) = (\dots, p_{k-1}, x_k, \mathsf{ap}_g(w) \cdot q_k, \dots) \qquad (\text{for } w : \Pi_1 A(x_k, x'_k))$$
$$(\dots, q_k \cdot \mathsf{ap}_g(w), y'_k, p_k, \dots) = (\dots, q_k, y_k, \mathsf{ap}_f(w) \cdot p_k, \dots). \qquad (\text{for } w : \Pi_1 A(y_k, y'_k))$$

However, these are not necessary! In fact, they follow automatically by path induction on w. This is the main difference between the "naive" Seifert–van Kampen theorem and the more refined one we will consider in Section 4.

3.2 The Encode-decode Proof

Before beginning the encode-decode proof proper, we characterize transports in the fibration code:

- For $p : b =_B b'$ and $u : P$, we have

$$\mathsf{transport}^{b \mapsto \mathsf{code}(u, i(b))}(p, (\dots, y_n, p_n, b)) = (\dots, y_n, p_n \cdot p, b').$$

- For $q : c =_C c'$ and $u : P$, we have

$$\mathsf{transport}^{c \mapsto \mathsf{code}(u, j(c))}(q, (\dots, x_n, q_n, c)) = (\dots, x_n, q_n \cdot q, c').$$

Here we are abusing notation by using the same name for a path in X and its image in $\Pi_1 X$. Note that transport in $\Pi_1 X$ is also given by concatenation with (the image of) a path. From this we can prove the above statements by induction on u. We also have:

- For $a : A$ and $u : P$,

$$\mathsf{transport}^{v \mapsto \mathsf{code}(u, v)}(h(a), (\dots, y_n, p_n, f(a))) = (\dots, y_n, p_n, a, \mathsf{refl}_{g(a)}, g(a)).$$

This follows essentially from the definition of code.

Now, as is often the case, the function encode will be defined by transporting a "reflexivity code" along a path. The reflexivity code $r : \prod_{(u:P)} \mathsf{code}(u, u)$ is defined by induction on u as follows:

$$r(i(b)) :\equiv (b, \mathsf{refl}_b, b)$$
$$r(j(c)) :\equiv (c, \mathsf{refl}_c, c)$$

and for $r(h(a))$ we take the composite path

$$\begin{aligned}(h(a), h(a))_*(f(a), \mathsf{refl}_{f(a)}, f(a)) &= (g(a), \mathsf{refl}_{g(a)}, a, \mathsf{refl}_{f(a)}, a, \mathsf{refl}_{g(a)}, g(a)) \\ &= (g(a), \mathsf{refl}_{g(a)}, g(a))\end{aligned}$$

where the first path is by the observation above about transporting in code, and the second is an instance of the set quotient relation used to define code.

▶ **Theorem 2** (Naive Seifert–van Kampen theorem)**.** *For all $u, v : P$ there is an equivalence*

$$\Pi_1 P(u, v) \simeq \mathsf{code}(u, v).$$

Proof. To define a function $\mathsf{encode} : \Pi_1 P(u, v) \to \mathsf{code}(u, v)$ it suffices to define a function $(u =_P v) \to \mathsf{code}(u, v)$, since $\mathsf{code}(u, v)$ is a set. We do this by transport:

$$\mathsf{encode}(p) :\equiv \mathsf{transport}^{v \mapsto \mathsf{code}(u, v)}(p, r(u)).$$

Now to define $\mathsf{decode} : \mathsf{code}(u, v) \to \Pi_1 P(u, v)$ we proceed as usual by induction on $u, v : P$. In each case for u and v, we apply i or j to all the paths p_k and q_k as appropriate and concatenate the results in P, using h to identify the endpoints. For instance, when $u \equiv i(b)$ and $v \equiv i(b')$, we define

$$\begin{aligned}&\mathsf{decode}(b, p_0, x_1, q_1, y_1, p_1, \dots, y_n, p_n, b') \\ &\qquad :\equiv (p_0) \cdot h(x_1) \cdot \mathsf{ap}_j(q_1) \cdot h(y_1)^{-1} \cdot \mathsf{ap}_i(p_1) \cdot \dots \cdot h(y_n)^{-1} \cdot \mathsf{ap}_i(p_n). \quad (3)\end{aligned}$$

This respects the set-quotient equivalence relation and the equivalences such as (1), by the naturality and functoriality of paths [18, Chapter 2].

As usual with the encode-decode method, to show that the composite

$$\Pi_1 P(u, v) \xrightarrow{\mathsf{encode}} \mathsf{code}(u, v) \xrightarrow{\mathsf{decode}} \Pi_1 P(u, v)$$

is the identity, we first peel off the 0-truncation (since the codomain is a set) and then apply path induction. The input refl_u goes to $r(u)$, which then goes back to refl_u (applying a further induction on u to decompose $\mathsf{decode}(r(u))$).

Finally, consider the composite

$$\mathsf{code}(u,v) \xrightarrow{\mathsf{decode}} \Pi_1 P(u,v) \xrightarrow{\mathsf{encode}} \mathsf{code}(u,v).$$

We proceed by induction on $u, v : P$. When $u \equiv i(b)$ and $v \equiv i(b')$, this composite is

$$\begin{aligned}
(b, p_0, x_1, q_1, y_1, p_1, \dots, y_n, p_n, b') &\mapsto \Big(\mathsf{ap}_i(p_0) \cdot h(x_1) \cdot \mathsf{ap}_j(q_1) \cdot h(y_1)^{-1} \cdot \mathsf{ap}_i(p_1) \cdot \\
&\qquad \cdots \cdot h(y_n)^{-1} \cdot \mathsf{ap}_i(p_n)\Big)_* (r(i(b))) \\
&= \mathsf{ap}_i(p_n)_* \cdots \mathsf{ap}_j(q_1)_* h(x_1)_* \mathsf{ap}_i(p_0)_* (b, \mathsf{refl}_b, b) \\
&= \mathsf{ap}_i(p_n)_* \cdots \mathsf{ap}_j(q_1)_* h(x_1)_* (b, p_0, i(f(x_1))) \\
&= \mathsf{ap}_i(p_n)_* \cdots \mathsf{ap}_j(q_1)_* (b, p_0, x_1, \mathsf{refl}_{g(x_1)}, j(g(x_1))) \\
&= \mathsf{ap}_i(p_n)_* \cdots (b, p_0, x_1, q_1, j(g(y_1))) \\
&= \quad \vdots \\
&= (b, p_0, x_1, q_1, y_1, p_1, \dots, y_n, p_n, b').
\end{aligned}$$

i.e., the identity function. (To be precise, there is an implicit inductive argument needed here.) The other three point cases are analogous, and the path cases are trivial since all the types are sets. ◀

3.3 Examples

Theorem 2 allows us to calculate the fundamental groups of many types, provided A is a set, for in that case, each $\mathsf{code}(u,v)$ is, by definition, a set-quotient of a *set* by a relation. (This is because all of its ingredients belong to sets: the intermediate points $x_k : A$ and $y_k : A$ and the elements of truncated path spaces $\Pi_1 B$ and $\Pi_1 C$. We did notate b and b' in the definition of $\mathsf{code}(i(b), i(b'))$ in order to "anchor" the list on both sides, and B may not be a set; but for fixed endpoints $u = i(b)$, $v = i(b')$ these points are not allowed to vary, so they do not prevent $\mathsf{code}(u,v)$ from being a set.)

▶ **Example 3.** Let $A :\equiv \mathbf{2}$ be the 2-element type (the booleans), and let $B :\equiv \mathbf{1}$ and $C :\equiv \mathbf{1}$ be the 1-element type. Then P is equivalent to the circle $\mathbb{S}^1$. Inspecting the definition of, say, $\mathsf{code}(i(\star), i(\star))$, we see that the paths all may as well be trivial, so the only information is in the sequence of elements $x_1, y_1, \dots, x_n, y_n : \mathbf{2}$. Moreover, if we have $x_k = y_k$ or $y_k = x_{k+1}$ for any k, then the set-quotient relations allow us to excise both of those elements. Thus, every such sequence is identified with a canonical *reduced* one in which no two adjacent elements are equal. Clearly such a reduced sequence is uniquely determined by its length (a natural number n) together with, if $n > 1$, the information of whether x_1 is $0_{\mathbf{2}}$ or $1_{\mathbf{2}}$, since that determines the rest of the sequence uniquely. And these data can, of course, be identified with an integer, where n is the absolute value and x_1 encodes the sign. Thus we recover $\pi_1(\mathbb{S}^1) \cong \mathbb{Z}$ [15].

Since Theorem 2 asserts only a bijection of families of sets, this isomorphism $\pi_1(\mathbb{S}^1) \cong \mathbb{Z}$ is likewise only a bijection of sets. We could, however, define a concatenation operation on code (by concatenating sequences) and show that encode and decode form an isomorphism respecting this structure (i.e. an equivalence of groupoids, or "pregroupoids"). We leave the details to the reader.

▶ **Example 4.** Let $A :\equiv \mathbf{1}$ and B and C be arbitrary, so that f and g simply equip B and C with basepoints b and c, say. Then P is the *wedge* $B \vee C$ of B and C (the coproduct in

the category of based spaces). In this case, it is the elements x_k and y_k which are trivial, so that the only information is a sequence of loops $(p_0, q_1, p_1, \dots, p_n)$ with $p_k : \pi_1(B, b)$ and $q_k : \pi_1(C, c)$. Such sequences, modulo the equivalence relation we have imposed, are easily identified with the usual explicit description of the *free product* of the groups $\pi_1(B, b)$ and $\pi_1(C, c)$. Thus, $\pi_1(B \vee C)$ is isomorphic to this free product $\pi_1(B) * \pi_1(C)$.

Theorem 2 is also applicable to some cases when A is not a set, such as the following generalization of Example 3:

▶ **Example 5.** Let $B :\equiv \mathbf{1}$ and $C :\equiv \mathbf{1}$ but A be arbitrary; then P is, by definition, the *suspension* ΣA of A. Then once again the paths p_k and q_k are trivial, so that the only information in a path code is a sequence of elements $x_1, y_1, \dots, x_n, y_n : A$. The first two generating paths say that adjacent equal elements can be canceled, so it makes sense to think of this sequence as a word of the form $x_1 y_1^{-1} x_2 y_2^{-1} \cdots x_n y_n^{-1}$ in a group. Indeed, it looks similar to the free group on A (or equivalently on $\|A\|_0$), but we are considering only words that start with a non-inverted element, alternate between inverted and non-inverted elements, and end with an inverted one. This effectively reduces the size of the generating set by one. For instance, if A has a point $a : A$, then we can identify $\pi_1(\Sigma A)$ with the group presented by $\|A\|_0$ as generators with the relation $|a|_0 = e$.

In particular, if A is connected (that is, $\|A\|_0$ is contractible), it follows that $\pi_1(\Sigma A)$ is trivial. Since the higher spheres can be defined as $\mathbb{S}^{n+1} :\equiv \Sigma \mathbb{S}^n$, and $\mathbb{S}^1$ is easily seen to be connected, it follows that $\pi_1(\mathbb{S}^n) = 1$ for all $n > 1$.

However, Theorem 2 stops just short of being the full classical Seifert–van Kampen theorem, which we recall states that

$$\pi_1(B \sqcup^A C) \cong \pi_1(B) *_{\pi_1(A)} \pi_1(C) \tag{4}$$

(with base point coming from A). Specifically, when A is not a set, the paths in A should be able to "move back and forth" between the images of B and C in the pushout; but the definition of code, and hence the conclusion of Theorem 2, says nothing at all about $\pi_1(A)$! This "moving back and forth" still happens, of course, but it happens quietly by way of type dependency and transport: the paths such as p_k and q_k in $\mathsf{code}(u, v)$ depend on the intermediate points $x_k, y_k : A$, and hence can be transported forwards and backwards along paths in A. These transported paths then get collapsed in the set-quotient that defines $\mathsf{code}(u, v)$. So the quotienting involved in the "amalgamated free product" (4) still happens, but it happens in an "automatic" type-theoretic way that, while easier to define, makes it hard to extract explicit information. For this reason, we now consider a better version of the Seifert–van Kampen theorem.

4 Improvement with an Indexing Space

The improvement of Seifert–van Kampen we present now is closely analogous to a similar improvement in classical algebraic topology, where A is equipped with a *set S of base points*. In fact, it turns out to be unnecessary for our proof to assume that the "set of basepoints" is a *set* — it might just as well be an arbitrary type. The utility of assuming S is a set arises later, when applying the theorem to obtain computations. What is important is that S contains at least one point in each connected component of A. We state this in type theory by saying that we have a type S and a function $\kappa : S \to A$ which is surjective, i.e. (-1)-connected. If $S \equiv A$ and κ is the identity function, then we will recover Theorem 2.

Another example to keep in mind is when A is pointed and (0-)connected, with $\kappa : \mathbf{1} \to A$ the point: by [18, Lemmas 7.5.2 and 7.5.11] this map is surjective just when A is 0-connected.

Let $A, B, C, f, g, P, i, j, h$ be as in the previous section. We now define, given our surjective map $\kappa : S \to A$, an auxiliary type which improves the connectedness of κ. Let T be the higher inductive type generated by

- A function $\ell : S \to T$, and
- For each $s, s' : S$, a function $m : (\kappa(s) =_A \kappa(s')) \to (\ell(s) =_T \ell(s'))$.

There is an obvious induced function $\overline{\kappa} : T \to A$ such that $\overline{\kappa} \circ \ell = \kappa$, and any $p : \kappa(s) = \kappa(s')$ is identified with the composite $\kappa(s) = \overline{\kappa}(\ell(s)) \overset{\overline{\kappa}(m(p))}{=} \overline{\kappa}(\ell(s')) = \kappa(s')$.

▶ **Lemma 6.** *$\overline{\kappa}$ is 0-connected.*

Proof. We must show that for all $a : A$, the 0-truncation of the type $\sum_{(t:T)}(\overline{\kappa}(t) = a)$ is contractible. Since contractibility is a mere proposition and κ is (-1)-connected, we may assume that $a = \kappa(s)$ for some $s : S$. Now we can take the center of contraction to be $|(\ell(s), q)|_0$ where q is the path $\overline{\kappa}(\ell(s)) = \kappa(s)$.

It remains to show that for any $\phi : \left\|\sum_{(t:T)}(\overline{\kappa}(t) = \kappa(s))\right\|_0$ we have $\phi = |(\ell(s), q)|_0$. Since the latter is a mere proposition, and in particular a set, we may assume that $\phi = |(t, p)|_0$ for $t : T$ and $p : \overline{\kappa}(t) = \kappa(s)$.

Now we can do induction on $t : T$. If $t \equiv \ell(s')$, then $\kappa(s') = \overline{\kappa}(\ell(s')) \overset{p}{=} \kappa(s)$ yields via m a path $\ell(s) = \ell(s')$. Hence by definition of $\overline{\kappa}$ and of identification in homotopy fibers, we obtain a path $(\kappa(s'), p) = (\kappa(s), q)$, and thus $|(\kappa(s'), p)|_0 = |(\kappa(s), q)|_0$. Next we must show that as t varies along m these paths agree. But they are paths in a set (namely $\left\|\sum_{(t:T)}(\overline{\kappa}(t) = \kappa(s))\right\|_0$), and hence this is automatic. ◀

▶ Remark 7. T can be regarded as the (homotopy) coequalizer of the "kernel pair" of κ. If S and A were sets, then the (-1)-connectivity of κ would imply that A is the 0-truncation of this coequalizer (this is a standard fact about exact categories, proven in our context in [18, Chapter 10]). For general types, higher topos theory suggests that (-1)-connectivity of κ will imply instead that A is the colimit (a.k.a. "geometric realization") of the "simplicial kernel" of κ. The type T is the colimit of the "1-skeleton" of this simplicial kernel, so it makes sense that it improves the connectivity of κ by 1. More generally, we might expect the colimit of the n-skeleton to improve connectivity by n.

4.1 New code

Now we define $\mathsf{code} : P \to P \to \mathcal{U}$ by double induction as follows.

- $\mathsf{code}(i(b), i(b'))$ is now a set-quotient of the type of sequences

$$(b, p_0, x_1, q_1, y_1, p_1, x_2, q_2, y_2, p_2, \dots, y_n, p_n, b')$$

where

- $n : \mathbb{N}$,
- $x_k : S$ and $y_k : S$ for $0 < k \leq n$,
- $p_0 : \Pi_1 B(b, f(\kappa(x_1)))$ and $p_n : \Pi_1 B(f(\kappa(y_n)), b')$ for $n > 0$, and $p_0 : \Pi_1 B(b, b')$ for $n = 0$,
- $p_k : \Pi_1 B(f(\kappa(y_k)), f(\kappa(x_{k+1})))$ for $1 \leq k < n$,
- $q_k : \Pi_1 C(g(\kappa(x_k)), g(\kappa(y_k)))$ for $1 \leq k \leq n$.

The quotient is generated by the following paths, as before:

$$(\dots, q_k, y_k, \mathsf{refl}_{f(y_k)}, y_k, q_{k+1}, \dots) = (\dots, q_k \cdot q_{k+1}, \dots)$$
$$(\dots, p_k, x_k, \mathsf{refl}_{g(x_k)}, x_k, p_{k+1}, \dots) = (\dots, p_k \cdot p_{k+1}, \dots)$$

and also the following new paths (see Remark 1):

$$(\dots, p_{k-1} \cdot \mathsf{ap}_f(w), x'_k, q_k, \dots) = (\dots, p_{k-1}, x_k, \mathsf{ap}_g(w) \cdot q_k, \dots) \quad (\text{for } w : \Pi_1 A(\kappa(x_k), \kappa(x'_k)))$$
$$(\dots, q_k \cdot \mathsf{ap}_g(w), y'_k, p_k, \dots) = (\dots, q_k, y_k, \mathsf{ap}_f(w) \cdot p_k, \dots) \quad (\text{for } w : \Pi_1 A(\kappa(y_k), \kappa(y'_k))).$$

We will need below the definition of the case of decode on such a sequence, which as before concatenates all the paths p_k and q_k together with instances of h to give an element of $\Pi_1 P(i(f(b)), i(f(b')))$, cf. (3). As before, the other three point cases are nearly identical.

- For $a : A$ and $b : B$, we require an equivalence

$$\mathsf{code}(i(b), i(f(a))) \simeq \mathsf{code}(i(b), j(g(a))). \tag{5}$$

Since code is set-valued, by Theorem 6 we may assume that $a = \overline{\kappa}(t)$ for some $t : T$. Next, we can do induction on t. If $t \equiv \ell(s)$ for $s : S$, then we define (5) as in Section 3:

$$(\dots, y_n, p_n, f(\kappa(s))) \mapsto (\dots, y_n, p_n, s, \mathsf{refl}_{g(\kappa(s))}, g(\kappa(s)))$$
$$(\dots, x_n, p_n, s, \mathsf{refl}_{f(\kappa(s))}, f(\kappa(s))) \mapsfrom (\dots, x_n, p_n, g(\kappa(s))).$$

These respect the equivalence relations, and define quasi-inverses just as before. Now suppose t varies along $\mathsf{ap}_{m_{s,s'}}(w)$ for some $w : \kappa(s) = \kappa(s')$; we must show that (5) respects transporting along $\mathsf{ap}_{\overline{\kappa}}(\mathsf{ap}_m(w))$. By definition of $\overline{\kappa}$, this essentially boils down to transporting along w itself. By the characterization of transport in path types, what we need to show is that

$$w_*(\dots, y_n, p_n, f(\kappa(s))) = (\dots, y_n, p_n \cdot \mathsf{ap}_f(w), f(\kappa(s')))$$

is mapped by (5) to

$$w_*(\dots, y_n, p_n, s, \mathsf{refl}_{g(\kappa(s))}, g(\kappa(s))) = (\dots, y_n, p_n, s, \mathsf{refl}_{g(\kappa(s))} \cdot \mathsf{ap}_g(w), g(\kappa(s'))).$$

But this follows directly from the new generators we have imposed on the set-quotient relation defining code.

- The other three requisite equivalences are defined similarly.
- Finally, since the commutativity (2) is a mere proposition, by (-1)-connectedness of κ we may assume that $a = \kappa(s)$ and $a' = \kappa(s')$, in which case it follows exactly as before.

4.2 Improved Theorem

▶ **Theorem 8** (Seifert–van Kampen with a set of basepoints). *For all $u, v : P$ there is an equivalence $\Pi_1 P(u, v) \simeq \mathsf{code}(u, v)$ with code defined as in Section 4.1.*

Proof. Basically just like before. To show that decode respects the new generators of the quotient relation, we use the naturality of h. And to show that decode respects the equivalences such as (5), we need to induct on $\overline{\kappa}$ and on T in order to decompose those equivalences into their definitions, but then it becomes again simply functoriality of f and g. The rest is easy. In particular, no additional argument is required for $\mathsf{encode} \circ \mathsf{decode}$, since the goal is to prove an equality in a set, and so the case of h is trivial. ◀

4.3 Examples

Theorem 8 allows us to calculate the fundamental group of a pushout $B \sqcup^A C$ even when A is not a set, provided S is a set, for in that case, each $\mathsf{code}(u, v)$ is, by definition, a set-quotient of a *set* by a relation. In that respect, it is an improvement over Theorem 2.

▶ **Example 9.** Suppose $S :\equiv \mathbf{1}$, so that A has a basepoint $a :\equiv \kappa(\star)$ and is connected. Then code for loops in the pushout can be identified with alternating sequences of loops in $\pi_1(B, f(a))$ and $\pi_1(C, g(a))$, modulo an equivalence relation which allows us to slide elements of $\pi_1(A, a)$ between them (after applying f and g respectively). Thus, $\pi_1(P)$ can be identified with the *amalgamated free product* $\pi_1(B) *_{\pi_1(A)} \pi_1(C)$ (the pushout in the category of groups). This (in the case when B and C are open subspaces of the pushout P, and A is their intersection) is probably the most classical version of the Seifert–van Kampen theorem.

▶ **Example 10.** As a special case of Example 9, suppose additionally that $C :\equiv \mathbf{1}$, so that P is the cofiber B/A. Then every loop in C is identified with reflexivity, so the relations on path codes allow us to collapse all sequences to a single loop in B. The additional relations require that multiplying on the left, right, or in the middle by an element in the image of $\pi_1(A)$ is the identity. We can thus identify $\pi_1(B/A)$ with the quotient of the group $\pi_1(B)$ by the normal subgroup generated by the image of $\pi_1(A)$.

▶ **Example 11.** As a further special case of Example 10, let $B :\equiv \mathbb{S}^1 \vee \mathbb{S}^1$, let $A :\equiv \mathbb{S}^1$, and let $f : A \to B$ pick out the composite loop $p \cdot q \cdot p^{-1} \cdot q^{-1}$, where p and q are the generating loops in the two copies of $\mathbb{S}^1$ comprising B. Then P is a presentation of the torus T^2 [13]. Thus, $\pi_1(T^2)$ is the quotient of the free group on two generators (i.e., $\pi_1(B)$) by the relation $p \cdot q \cdot p^{-1} \cdot q^{-1} = 1$. This clearly yields the free *abelian* group on two generators, which is $\mathbb{Z} \times \mathbb{Z}$.

▶ **Example 12.** More generally, any CW complex can be obtained by repeatedly "coning off" spheres. That is, we start with a set X_0 of points ("0-cells"), which is the "0-skeleton" of the CW complex. We take the pushout

$$\begin{array}{ccc} S_1 \times \mathbb{S}^0 & \xrightarrow{f_1} & X_0 \\ \downarrow & & \downarrow \\ \mathbf{1} & \longrightarrow & X_1 \end{array}$$

for some set S_1 of 1-cells and some family f_1 of "attaching maps", obtaining the "1-skeleton" X_1. Then we take the pushout

$$\begin{array}{ccc} S_2 \times \mathbb{S}^1 & \xrightarrow{f_2} & X_1 \\ \downarrow & & \downarrow \\ \mathbf{1} & \longrightarrow & X_2 \end{array}$$

for some set S_2 of 2-cells and some family f_2 of attaching maps, obtaining the 2-skeleton X_2, and so on. The fundamental group of each pushout can be calculated from the Seifert–van Kampen theorem: we obtain the group presented by generators derived from the 1-skeleton, and relations derived from S_2 and f_2. The pushouts after this stage do not alter the fundamental group, since (as noted in Example 5) $\pi_1(\mathbb{S}^n)$ is trivial for $n > 1$.

▶ **Example 13.** In particular, suppose given any presentation of a group $G = \langle X \mid R \rangle$, with X a set of generators and R a set of words in these generators. Let $B :\equiv \bigvee_X \mathbb{S}^1$ and $A :\equiv \bigvee_R \mathbb{S}^1$, with $f : A \to B$ sending each copy of $\mathbb{S}^1$ to the corresponding word in the generating loops of B. It follows that $\pi_1(P) \cong G$; thus we have constructed a connected type whose fundamental group is G. Since any group G has a presentation (e.g. let $X = G$ and R the set of all products that equal the identity), any group is the fundamental group of some type. If we 1-truncate such a type, we obtain a type whose only nontrivial homotopy group is G; this is called an **Eilenberg–Mac Lane space** $K(G, 1)$. (Eilenberg–Mac Lane spaces in homotopy type theory were constructed more explicitly by [14]).

5 Mechanization

Overall, the structure of the Agda proof follows closely the informal argument, giving evidence that homotopy type theory is suitable for mechanization. The major difference from the informal proof is that homotopy pushouts and set quotients are simulated due to the lack of native support. Although there is a trick due to [11], which we use, that enables some of the computation rules to be definitional (those involving point constructors), many computations still need to be manually carried out, resulting in 900 lines of Agda code merely for the well-definedness of the type family code. (The entire mechanization is of roughly 1,800 lines.)

Homotopy pushouts and set quotients can be easily defined as higher inductive types. Unfortunately, Agda like other proof assistants such as Coq [5] was designed for a more traditional variant of Martin-Löf type theory without the univalence axiom and higher inductive types. Lean [6] has built-in quotients and truncations but still only computes on point constructors. The proof assistant Cubical [3] aims to restore full computation rules but was not mature enough at the time of this mechanization.

Finally, the cubical approach [13] is shown to be useful in handling coherence conditions in many theorems, for example the homotopy groups of torus [13], the 3×3 lemma [13], and the Mayer-Vietoris sequences [4]. This might also simplify the proof of Eq. (2) while constructing the family code.

References

1. Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science*, 25:1010–1039, 6 2015. doi:10.1017/S0960129514000486.
2. Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, jan 2009. doi:10.1017/S0305004108001783.
3. Marc Bezem, Thierry Coquand, and Simon Huber. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs*, volume 26 of *Leibniz International Proceedings in Informatics*, pages 107–128. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.TYPES.2013.107.
4. Evan Cavallo. The Mayer-Vietoris sequence in HoTT. In *Oxford Quantum Video*. The QMAC and Clay Mathematics Institute and The EPSRC, 2014. URL: https://youtu.be/6QCFV4op1Uo.
5. The Coq proof assistant. URL: https://coq.inria.fr/.
6. Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *Automated Deduction – CADE-*

25, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015. doi:10.1007/978-3-319-21401-6_26.

7 Nicola Gambino and Richard Garner. The identity type weak factorisation system. *Theoretical Computer Science*, 409(1):94–109, 2008. doi:10.1016/j.tcs.2008.08.030.

8 Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford University Press, 1998. arXiv:pLnKggT_In4C.

9 Kuen-Bang Hou (Favonia), Eric Finster, Daniel R. Licata, and Peter LeFanu Lumsdaine. A mechanization of the Blakers-Massey connectivity theorem in homotopy type theory, May 2016. arXiv:1605.03227.

10 Chris Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after Voevodsky), 2012. arXiv:1211.2851.

11 Daniel R. Licata. Running circles around (in) your proof assistant; or, quotients that compute, 2011. URL: http://homotopytypetheory.org/2011/04/23/running-circles-around-in-your-proof-assistant/.

12 Daniel R. Licata and Guillaume Brunerie. $\pi_n(S^n)$ in homotopy type theory. *CPP*, 2013. URL: http://dlicata.web.wesleyan.edu/pubs/lb13cpp/lb13cpp.pdf.

13 Daniel R. Licata and Guillaume Brunerie. A cubical approach to synthetic homotopy theory. *LICS*, 2015. URL: http://dlicata.web.wesleyan.edu/pubs/lb15cubicalsynth/lb15cubicalsynth.pdf.

14 Daniel R. Licata and Eric Finster. Eilenberg–MacLane spaces in homotopy type theory. *LICS*, 2014. URL: http://dlicata.web.wesleyan.edu/pubs/lf14em/lf14em.pdf.

15 Daniel R. Licata and Michael Shulman. Calculating the fundamental group of the circle in homotopy type theory. In *LICS'13*, 2013. arXiv:1301.3443.

16 Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007. URL: http://www.cse.chalmers.se/~ulfn/papers/thesis.html.

17 Charles Rezk. Proof of the Blakers–Massey theorem, 2014. URL: http://www.math.uiuc.edu/~rezk/freudenthal-and-blakers-massey.pdf.

18 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations for Mathematics*. Institute for Advanced Study, git commit hash g662cdd8 edition, 2013. URL: http://homotopytypetheory.org/book.

19 Benno van den Berg and Richard Garner. Topological and simplicial models of identity types. *ACM Transactions on Computational Logic*, 13(1):3:1–3:44, 2012. doi:10.1145/2071368.2071371.

20 Michael A. Warren. *Homotopy theoretic aspects of constructive type theory*. PhD thesis, Carnegie Mellon University, 2008. URL: http://mawarren.net/papers/phd.pdf.

Guarded Cubical Type Theory: Path Equality for Guarded Recursion

Lars Birkedal[1], Aleš Bizjak[2], Ranald Clouston[3], Hans Bugge Grathwohl[4], Bas Spitters[5], and Andrea Vezzosi[6]

1 **Department of Computer Science, Aarhus University, Denmark**
2 **Department of Computer Science, Aarhus University, Denmark**
3 **Department of Computer Science, Aarhus University, Denmark**
4 **Department of Computer Science, Aarhus University, Denmark**
5 **Department of Computer Science, Aarhus University, Denmark**
6 **Department of Computer Science and Engineering, Chalmers University of Technology, Sweden**

Abstract

This paper improves the treatment of equality in guarded dependent type theory (GDTT), by combining it with cubical type theory (CTT). GDTT is an extensional type theory with guarded recursive types, which are useful for building models of program logics, and for programming and reasoning with coinductive types. We wish to implement GDTT with decidable type checking, while still supporting non-trivial equality proofs that reason about the extensions of guarded recursive constructions. CTT is a variation of Martin-Löf type theory in which the identity type is replaced by abstract paths between terms. CTT provides a computational interpretation of functional extensionality, is conjectured to have decidable type checking, and has an implemented type checker. Our new type theory, called guarded cubical type theory, provides a computational interpretation of extensionality for guarded recursive types. This further expands the foundations of CTT as a basis for formalisation in mathematics and computer science. We present examples to demonstrate the expressivity of our type theory, all of which have been checked using a prototype type-checker implementation, and present semantics in a presheaf category.

1998 ACM Subject Classification F.3.3 Studies of Program Constructs, F.3.2 Semantics of Programming Languages

Keywords and phrases Guarded Recursion, Dependent Type Theory, Cubical Type Theory, Denotational Semantics, Homotopy Type Theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.23

1 Introduction

Guarded recursion is a technique for defining and reasoning about infinite objects. Its applications include the definition of productive operations on data structures more commonly defined via coinduction, such as streams, and the construction of models of program logics for modern programming languages with features such as higher-order store and concurrency [6]. This is done via the type-former $\triangleright$, called 'later', which distinguishes data which is available immediately from data only available after some computation, such as the unfolding of a fixed-point. For example, guarded recursive streams are defined by the equation

$$\mathsf{Str}_A \;=\; A \times \triangleright \mathsf{Str}_A$$

rather than the more standard $\mathsf{Str}_A = A \times \mathsf{Str}_A$, to specify that the head is available now but the tail only later. The type for fixed-point combinators is then $(\triangleright A \to A) \to A$, rather

© Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi;
licensed under Creative Commons License CC-BY

than the logically inconsistent $(A \to A) \to A$, disallowing unproductive definitions such as taking the fixed-point of the identity function.

Guarded recursive types were developed in a simply-typed setting by Clouston et al. [9], following earlier work [21, 3, 1], alongside a logic for reasoning about such programs. For large examples such as models of program logics, we would like to be able to formalise such reasoning. A major approach to formalisation is via *dependent types*, used for example in the proof assistants Coq [18] and Agda [22]. Bizjak et al. [8], following earlier work [5, 20], introduced guarded dependent type theory (GDTT), integrating the $\triangleright$ type-former into a dependently typed calculus, and supporting the definition of guarded recursive types as fixed-points of functions on universes, and guarded recursive operations on these types.

We wish to formalise non-trivial theorems about equality between guarded recursive constructions, but such arguments often cannot be accommodated within *intensional* Martin-Löf type theory. For example, we may need to be able to reason about the extensions of streams in order to prove the equality of different stream functions. Hence GDTT includes an equality reflection rule, which is well known to make type checking undecidable. This problem is close to well-known problems with functional extensionality [13, Sec. 3.1.3], and indeed this analogy can be developed. Just as functional extensionality involves mapping terms of type $(x : A) \to \mathsf{Id}\, B\, (f\, x)\, (g\, x)$ to proofs of $\mathsf{Id}\, (A \to B)\, f\, g$, extensionality for guarded recursion requires an extensionality principle for later types, namely the ability to map terms of type $\triangleright\, \mathsf{Id}\, A\, t\, u$ to proofs of $\mathsf{Id}\, (\triangleright A)\, (\mathsf{next}\, t)\, (\mathsf{next}\, u)$, where next is the constructor for $\triangleright$. These types are isomorphic in the intended model, the presheaf category $\widehat{\omega}$ known as the *topos of trees*, and so in GDTT their equality was asserted as an axiom. But in a calculus without equality reflection we cannot merely assert such axioms without losing canonicity.

Cubical type theory (CTT) [10] is a new type theory with a computational interpretation of functional extensionality but without equality reflection, and hence is a candidate for extension with guarded recursion, so that we may formalise our arguments without incurring the disadvantages of fully extensional identity types. CTT was developed primarily to provide a computational interpretation of the univalence axiom of Homotopy Type Theory [26]. The most important novelty of CTT is the replacement of inductively defined identity types by *paths*, which can be seen as maps from an abstract interval $\mathbb{I}$, and are introduced and eliminated much like functions. CTT can be extended with identity types which model all rules of standard Martin-Löf type theory [10, Sec. 9.1], but these are equivalent to path types, and in our paper it suffices to work with path types only. CTT has sound denotational semantics in (fibrations in) *cubical sets*, a presheaf category that is used to model homotopy types. Many basic syntactic properties of CTT, such as the decidability of type checking, and canonicity for base types, are yet to be proved, but a type checker has been implemented [1] that confers some confidence in such properties.

In Sec. 2 of this paper we propose *guarded cubical type theory* (GCTT), a combination of the two type theories[2] which supports non-trivial proofs about guarded recursive types via path equality, while retaining the potential for good syntactic properties such as decidable type-checking and canonicity. In particular, just as a term can be defined in CTT to witness functional extensionality, a term can be defined in GCTT to witness extensionality for later types. Further, we use elements of the interval of CTT to annotate fixed-points, and hence control their unfoldings. This ensures that fixed-points are path equal, but not judgementally equal, to their unfoldings, and hence prevents infinite unfoldings, an obvious source of

[1] `https://github.com/mortberg/cubicaltt`

[2] With the exception of the *clock quantification* of GDTT, which we leave to future work.

non-termination in any calculus with infinite constructions. The resulting calculus is shown via examples to be useful for reasoning about guarded recursive operations; we also view it as potentially significant from the point of view of CTT, extending its expressivity as a basis for formalisation.

In Sec. 3 we give sound semantics to this type theory via the presheaf category over the product of the categories used to define semantics for GDTT and CTT. This requires considerable work to ensure that the constructions of the two type theories remain sound in the new category, particularly the glueing and universe of CTT. The key technical challenge is to ensure that the $\triangleright$ type-former supports the *compositions* that all types must carry in the semantics of CTT.

We have implemented a prototype type-checker for this extended type theory [3], which provides confidence in the type theory's syntactic properties. All examples in this paper, and many others, have been formalised in this type checker.

For reasons of space many details and proofs are omitted from this paper, but are included in a technical appendix[4].

2 Guarded Cubical Type Theory

This section introduces guarded cubical type theory (GCTT), and presents examples of how it can be used to prove properties of guarded recursive constructions.

2.1 Cubical Type Theory

We first give a brief overview of *cubical type theory* [5] (CTT) [10]. We start with a standard dependent type theory with Π, Σ, natural numbers, and a Russell-style universe:

$$
\begin{array}{lcll}
\Gamma, \Delta & ::= & () \mid \Gamma, x : A & \text{Contexts} \\
t, u, A, B & ::= & x \mid \lambda x : A.t \mid t\, u \mid (x : A) \to B & \Pi\text{-types} \\
& \mid & (t, u) \mid t.1 \mid t.2 \mid (x : A) \times B & \Sigma\text{-types} \\
& \mid & 0 \mid \mathsf{s}\, t \mid \mathsf{natrec}\, t\, u \mid \mathsf{N} & \text{Natural numbers} \\
& \mid & \mathsf{U} & \text{Universe}
\end{array}
$$

We adhere to the usual conventions of considering terms and types up to α-equality, and writing $A \to B$, respectively $A \times B$, for non-dependent Π and Σ-types. We use the symbol '$=$' for judgemental equality.

The central novelty of CTT is its treatment of equality. Instead of the inductively defined identity types of intensional Martin-Löf type theory [17], CTT has *paths*. The paths between two terms t, u of type A form a sort of function space, intuitively that of continuous maps from some interval $\mathbb{I}$ to A, with endpoints t and u. Rather than defining the interval $\mathbb{I}$ concretely as the unit interval $[0, 1] \subseteq \mathbb{R}$, it is defined as the *free De Morgan algebra* on a discrete infinite set of names $\{i, j, k, \dots\}$. A De Morgan algebra is a bounded distributive lattice with an involution $1 - \cdot$ satisfying the De Morgan laws

$$1 - (i \wedge j) = (1 - i) \vee (1 - j), \qquad 1 - (i \vee j) = (1 - i) \wedge (1 - j).$$

The interval $[0, 1] \subseteq \mathbb{R}$, with min, max and $1 - \cdot$, is an example of a De Morgan algebra.

[3] http://github.com/hansbugge/cubicaltt/tree/gcubical
[4] http://cs.au.dk/~birke/papers/gdtt-cubical-technical-appendix.pdf
[5] http://www.cse.chalmers.se/~coquand/selfcontained.pdf is a self-contained presentation of CTT.

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{Path}\ A\ t\ u}$$

$$\frac{\Gamma \vdash A \qquad \Gamma, i : \mathbb{I} \vdash t : A}{\Gamma \vdash \langle i \rangle\, t : \mathsf{Path}\ A\ t[0/i]\ t[1/i]} \qquad \frac{\Gamma \vdash t : \mathsf{Path}\ A\ u\ s \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash t\, r : A}$$

Figure 1 Typing rules for path types.

The syntax for elements of $\mathbb{I}$ is:

$$r, s \ ::= \ 0 \mid 1 \mid i \mid 1 - r \mid r \wedge s \mid r \vee s.$$

0 and 1 represent the endpoints of the interval. We extend the definition of contexts to allow introduction of a new name:

$$\Gamma, \Delta \ ::= \ \cdots \mid \Gamma, i : \mathbb{I}.$$

The judgement $\Gamma \vdash r : \mathbb{I}$ means that r draws its names from Γ. Despite this notation, $\mathbb{I}$ is not a first-class type. Path types and their elements are defined by the rules in Fig. 1. *Path abstraction*, $\langle i \rangle\, t$, and *path application*, $t\, r$, are analogous to λ-abstraction and function application, and support the familiar β-equality $(\langle i \rangle\, t)\, r = t[r/i]$ and η-equality $\langle i \rangle\, t\, i = t$. There are two additional judgemental equalities for paths, regarding their endpoints: given $p : \mathsf{Path}\ A\ t\ u$ we have $p\, 0 = t$ and $p\, 1 = u$.

Paths provide a notion of identity which is more extensional than that of intensional Martin-Löf identity types, as exemplified by the proof term for functional extensionality:

$$\mathsf{funext}\ f\ g \triangleq \lambda p.\, \langle i \rangle\, \lambda x.\, p\, x\, i \ : \ ((x : A) \to \mathsf{Path}\, B\ (f\, x)\ (g\, x)) \to \mathsf{Path}\ (A \to B)\ f\ g.$$

The rules above suffice to ensure that path equality is reflexive, symmetric, and a congruence, but we also need it to be transitive and, where the underlying type is the universe, to support a notion of transport. This is done via *(Kan) composition operations*.

To define these we need the *face lattice*, $\mathbb{F}$, defined as the free distributive lattice on the symbols $(i = 0)$ and $(i = 1)$ for all names i, quotiented by the relation $(i = 0) \wedge (i = 1) = 0_{\mathbb{F}}$. The syntax for elements of $\mathbb{F}$ is:

$$\varphi, \psi \ ::= \ 0_{\mathbb{F}} \mid 1_{\mathbb{F}} \mid (i = 0) \mid (i = 1) \mid \varphi \wedge \psi \mid \varphi \vee \psi.$$

As with the interval, $\mathbb{F}$ is not a first-class type, but the judgement $\Gamma \vdash \varphi : \mathbb{F}$ asserts that φ draws its names from Γ. We also have the judgement $\Gamma \vdash \varphi = \psi : \mathbb{F}$ which asserts the equality of φ and ψ in the face lattice. Contexts can be restricted by elements of $\mathbb{F}$:

$$\Gamma, \Delta \ ::= \ \cdots \mid \Gamma, \varphi.$$

Such a restriction affects equality judgements so that, for example, $\Gamma, \varphi \vdash \psi_1 = \psi_2 : \mathbb{F}$ is equivalent to $\Gamma \vdash \varphi \wedge \psi_1 = \varphi \wedge \psi_2 : \mathbb{F}$

We write $\Gamma \vdash t : A[\varphi \mapsto u]$ as an abbreviation for the two judgements $\Gamma \vdash t : A$ and $\Gamma, \varphi \vdash t = u : A$, noting the restriction with φ in the equality judgement. Now the composition operator is defined by the typing and equality rule

$$\frac{\Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash a_0 : A[0/i][\varphi \mapsto u[0/i]]}{\Gamma \vdash \mathsf{comp}^i\ A\ [\varphi \mapsto u]\ a_0 : A[1/i][\varphi \mapsto u[1/i]]}.$$

A simple use of composition is to implement the transport operation for Path types

$$\mathsf{transp}^i\ A\ a\ \triangleq\ \mathsf{comp}^i\ A\ [0_{\mathbb{F}} \mapsto []]\ a\ :\ A[1/i],$$

where a has type $A[0/i]$. The notation [] stands for an empty *system*. In general a system is a list of pairs of faces and terms, and it defines an element of a type by giving the individual components at each face. We extend the syntax as follows:

$$t, u, A, B\ ::=\ \cdots \mid [\varphi_1\ t_1, \ldots, \varphi_n\ t_n].$$

Below we see two of the rules for systems; they ensure that the components of a system agree where the faces overlap, and that all the cases possible in the current context are covered:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash \varphi_1 \vee \ldots \vee \varphi_n = 1_{\mathbb{F}} : \mathbb{F} \qquad \Gamma, \varphi_i \vdash t_i : A \qquad \Gamma, \varphi_i \wedge \varphi_j \vdash t_i = t_j : A \qquad i, j = 1 \ldots n}{\Gamma \vdash [\varphi_1\ t_1, \ldots, \varphi_n\ t_n] : A}$$

$$\frac{\Gamma \vdash [\varphi_1\ t_1, \ldots, \varphi_n\ t_n] : A \qquad \Gamma \vdash \varphi_i = 1_{\mathbb{F}} : \mathbb{F}}{\Gamma \vdash [\varphi_1\ t_1, \ldots, \varphi_n\ t_n] = t_i : A}$$

We will shorten $[\varphi_1 \vee \ldots \vee \varphi_n \mapsto [\varphi_1\ t_1, \ldots, \varphi_n\ t_n]]$ to $[\varphi_1 \mapsto t_1, \ldots, \varphi_n \mapsto t_n]$.

A non-trivial example of the use of systems is the proof that Path is transitive; given $p : \mathsf{Path}\ A\ a\ b$ and $q : \mathsf{Path}\ A\ b\ c$ we can define

$$\mathsf{transitivity}\ p\, q \triangleq \langle i \rangle\, \mathsf{comp}^j\ A\ [(i = 0) \mapsto a, (i = 1) \mapsto q\, j]\ (p\, i)\ :\ \mathsf{Path}\ A\ a\ c.$$

This builds a path between the appropriate endpoints because we have the equalities $\mathsf{comp}^j\ A\ [1_{\mathbb{F}} \mapsto a]\ (p\, 0) = a$ and $\mathsf{comp}^j\ A\ [1_{\mathbb{F}} \mapsto q\, j]\ (p\, 1) = q\, 1 = c$.

For reasons of space we have omitted the descriptions of some features of CTT, such as glueing, and the further judgemental equalities for terms of the form $\mathsf{comp}^i\ A\ [\varphi \mapsto u]\ a_0$ that depend on the structure of A.

2.2 Later Types

In Fig. 3 we present the 'later' types of guarded dependent type theory (GDTT) [8], with judgemental equalities in Figs. 4 and 5. Note that we do not add any new equation for the interaction of compositions with $\triangleright$; such an equation would be necessary if we were to add the eliminator prev for $\triangleright$, but this extension (which involves clock quantifiers) is left to further work. We delay the presentation of the fixed-point operation until the next section.

The typing rules use the *delayed substitutions* of GDTT, as defined in Fig. 2. Delayed substitutions resemble Haskell-style do-notation, or a delayed form of let-binding. If we have a term $t : \triangleright A$, we cannot access its contents 'now', but if we are defining a type or term that itself has some part that is available 'later', then this part *should* be able to use the contents of t. Therefore delayed substitutions allow terms of type $\triangleright A$ to be unwrapped by $\triangleright$ and next. As observed by Bizjak et al. [8] these constructions generalise the *applicative functor* [19] structure of 'later' types, by the definitions $\mathsf{pure}\ t \triangleq \mathsf{next}\, t$, and $f \circledast t \triangleq \mathsf{next}\, [f' \leftarrow f, t' \leftarrow t]\, .\, f'\, t'$, as well as a generalisation of the $\circledast$ operation from simple functions to Π-types. We here make the new observation that delayed substitutions can express the function $\widehat{\triangleright} : \triangleright \mathsf{U} \to \mathsf{U}$, introduced by Birkedal and Møgelberg [4] to express guarded recursive types as fixed-points on universes, as $\lambda u. \triangleright [u' \leftarrow u].u'$; see for example the definition of streams in Sec. 2.4.

$$\frac{\Gamma \vdash}{\vdash \cdot : \Gamma \nrightarrow \cdot} \qquad \frac{\vdash \xi : \Gamma \nrightarrow \Gamma' \qquad \Gamma \vdash t : \triangleright \xi.A}{\vdash \xi\,[x \leftarrow t] : \Gamma \nrightarrow \Gamma', x : A}$$

Figure 2 Formation rules for delayed substitutions.

$$\frac{\Gamma, \Gamma' \vdash A \qquad \vdash \xi : \Gamma \nrightarrow \Gamma'}{\Gamma \vdash \triangleright \xi.A} \qquad \frac{\Gamma, \Gamma' \vdash A : \mathsf{U} \qquad \vdash \xi : \Gamma \nrightarrow \Gamma'}{\Gamma \vdash \triangleright \xi.A : \mathsf{U}}$$

$$\frac{\Gamma, \Gamma' \vdash t : A \qquad \vdash \xi : \Gamma \nrightarrow \Gamma'}{\Gamma \vdash \mathsf{next}\, \xi.\, t : \triangleright \xi.A}$$

Figure 3 Typing rules for later types.

$$\frac{\vdash \xi\,[x \leftarrow t] : \Gamma \nrightarrow \Gamma', x : B \qquad \Gamma, \Gamma' \vdash A}{\Gamma \vdash \triangleright \xi\,[x \leftarrow t]\,.A = \triangleright \xi.A}$$

$$\frac{\vdash \xi\,[x \leftarrow t, y \leftarrow u]\,\xi' : \Gamma \nrightarrow \Gamma', x : B, y : C, \Gamma'' \qquad \Gamma, \Gamma' \vdash C \qquad \Gamma, \Gamma', x : B, y : C, \Gamma'' \vdash A}{\Gamma \vdash \triangleright \xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.A = \triangleright \xi\,[y \leftarrow u, x \leftarrow t]\,\xi'.A}$$

$$\frac{\vdash \xi : \Gamma \nrightarrow \Gamma' \qquad \Gamma, \Gamma', x : B \vdash A \qquad \Gamma, \Gamma' \vdash t : B}{\Gamma \vdash \triangleright \xi\,[x \leftarrow \mathsf{next}\, \xi.\, t]\,.A = \triangleright \xi.A[t/x]}$$

Figure 4 Type equality rules for later types (congruence and equivalence rules are omitted).

$$\frac{\vdash \xi\,[x \leftarrow t] : \Gamma \nrightarrow \Gamma', x : B \qquad \Gamma, \Gamma' \vdash u : A}{\Gamma \vdash \mathsf{next}\, \xi\,[x \leftarrow t]\,.\, u = \mathsf{next}\, \xi.\, u : \triangleright \xi.A}$$

$$\frac{\vdash \xi\,[x \leftarrow t, y \leftarrow u]\,\xi' : \Gamma \nrightarrow \Gamma', x : B, y : C, \Gamma'' \qquad \Gamma, \Gamma' \vdash C \qquad \Gamma, \Gamma', x : B, y : C, \Gamma'' \vdash v : A}{\Gamma \vdash \mathsf{next}\, \xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.\, v = \mathsf{next}\, \xi\,[y \leftarrow u, x \leftarrow t]\,\xi'.\, v : \triangleright \xi\,[x \leftarrow t, y \leftarrow u]\,\xi'.A}$$

$$\frac{\vdash \xi : \Gamma \nrightarrow \Gamma' \qquad \Gamma, \Gamma', x : B \vdash u : A \qquad \Gamma, \Gamma' \vdash t : B}{\Gamma \vdash \mathsf{next}\, \xi\,[x \leftarrow \mathsf{next}\, \xi.\, t]\,.\, u = \mathsf{next}\, \xi.\, u[t/x][\triangleright \xi.A[t/x]]} \qquad \frac{\Gamma \vdash t : \triangleright \xi.A}{\Gamma \vdash \mathsf{next}\, \xi\,[x \leftarrow t]\,.\, x = t[\triangleright \xi.A]}$$

Figure 5 Term equality rules for later types. We omit congruence and equivalence rules, and the rules for terms of type U, which reflect the type equality rules of Fig. 4.

$$\frac{\Gamma \vdash r : \mathbb{I} \qquad \Gamma, x : \triangleright A \vdash t : A}{\Gamma \vdash \mathsf{dfix}^r\, x.t : \triangleright A} \qquad \frac{\Gamma, x : \triangleright A \vdash t : A}{\Gamma \vdash \mathsf{dfix}^1\, x.t = \mathsf{next}\, t[\mathsf{dfix}^0\, x.t/x][\triangleright A]}.$$

Figure 6 Typing and equality rules for the delayed fixed-point

▶ **Example 1.** In GDTT it is essential that we can convert terms of type $\triangleright\xi.\, \mathsf{Id}_A\ t\ u$ into terms of type $\mathsf{Id}_{\triangleright\xi.A}\,(\mathsf{next}\,\xi.\,t)\,(\mathsf{next}\,\xi.\,u)$, as it is essential for *Löb induction*, the technique of proof by guarded recursion where we assume $\triangleright p$, deduce p, and hence may conclude p with no assumptions. This is achieved in GDTT by postulating as an axiom the following judgemental equality:

$$\mathsf{Id}_{\triangleright\xi.A}\,(\mathsf{next}\,\xi.\,t)\,(\mathsf{next}\,\xi.\,u) \;=\; \triangleright\xi.\,\mathsf{Id}_A\ t\ u \tag{1}$$

A term from left-to-right of (1) can be defined using the J-eliminator for identity types, but the more useful direction is right-to-left, as proofs of equality by Löb induction involve assuming that we later have a path, then converting this into a path on later types. In fact in GCTT we can define a term with the desired type:

$$\lambda p.\langle i\rangle\, \mathsf{next}\,\xi[p' \leftarrow p].\, p'\ i \;:\; (\triangleright\xi.\,\mathsf{Path}\ A\ t\ u) \to \mathsf{Path}\,(\triangleright\xi.A)\,(\mathsf{next}\,\xi.\,t)\,(\mathsf{next}\,\xi.\,u). \tag{2}$$

Note the similarity of this term and type with that of funext, for functional extensionality, presented on page 4. Indeed we claim that (2) provides a computational interpretation of extensionality for later types.

2.3 Fixed Points

In this section we complete the presentation of GCTT by addressing fixed points. In GDTT there are fixed-point constructions $\mathsf{fix}\,x.t$ with the judgemental equality $\mathsf{fix}\,x.t = t[\mathsf{next}\,\mathsf{fix}\,x.t/x]$. In GCTT we want decidable type checking, including decidable judgemental equality, and so we cannot admit such an unrestricted unfolding rule. Our solution it that fixed points should not be judgementally equal to their unfoldings, but merely *path equal*. We achieve this by decorating the fixed-point combinator with an interval element which specifies the position on this path. The 0-endpoint of the path is the stuck fixed-point term, while the 1-endpoint is the same term unfolded once. However this threatens canonicity for base types: if we allow stuck fixed-points in our calculus, we could have stuck closed terms $\mathsf{fix}^i\, x.t$ inhabiting N. To avoid this, we introduce the *delayed* fixed-point combinator dfix, which produces a term 'later' instead of a term 'now'. Its typing rule, and notion of equality, is given in Fig. 6. We will write $\mathsf{fix}^r\, x.t$ for $t[\mathsf{dfix}^r\, x.t/x]$, $\mathsf{fix}\, x.t$ for $\mathsf{fix}^0\, x.t$, and $\mathsf{dfix}\, x.t$ for $\mathsf{dfix}^0\, x.t$.

▶ **Lemma 2** (Canonical unfold lemma). *For any term $\Gamma, x : \triangleright A \vdash t : A$ there is a path between $\mathsf{fix}\, x.t$ and $t[\mathsf{next}\,\mathsf{fix}\, x.t/x]$, given by the term $\langle i\rangle\, \mathsf{fix}^i\, x.t$.*

Transitivity of paths (via compositions) ensures that $\mathsf{fix}\, x.t$ is path equal to any number of fixed-point unfoldings of itself.

A term a of type A is said to be a *guarded fixed point* of a function $f : \triangleright A \to A$ if there is a path from a to $f(\mathsf{next}\, a)$.

▶ **Proposition 3** (Unique guarded fixed points). *Any guarded fixed-point a of a term $f : \triangleright A \to A$ is path equal to $\mathsf{fix}\, x.f\, x$.*

Proof. Given $p : \mathsf{Path}\, A\ a\ (f\,(\mathsf{next}\, a))$, we proceed by Löb induction, i.e., by assuming $\mathsf{ih} : \triangleright(\mathsf{Path}\, A\ a\ (\mathsf{fix}\, x.f\, x))$. We can define a path

$$s \triangleq \langle i \rangle\ f(\mathsf{next}\,[q \leftarrow \mathsf{ih}]\,.\,q\,i)\ :\ \mathsf{Path}\, A\ (f(\mathsf{next}\, a))\ (f(\mathsf{next}\,\mathsf{fix}\, x.f\, x)),$$

which is well-typed because the type of the variable q ensures that $q\,0$ is judgementally equal to a, resp. $q\,1$ and $\mathsf{fix}\, x.f\, x$. Note that we here implicitly use the extensionality principle for later (2). We compose s with p, and then with the inverse of the canonical unfold lemma of Lem. 2, to obtain our path from a to $\mathsf{fix}\, x.f\, x$. We can write out our full proof term, where p^{-1} is the inverse path of p, as

$$\mathsf{fix}\,\mathsf{ih}\,.\,\langle i \rangle\,\mathsf{comp}^j\ A\ [(i=0) \mapsto p^{-1}, (i=1) \mapsto f(\mathsf{dfix}^{1-j}\, x.f\, x)]\ (f(\mathsf{next}\,[q \leftarrow \mathsf{ih}]\,.\,q\,i)).$$ ◀

2.4 Programming and Proving with Guarded Recursive Types

In this section we show some simple examples of programming with guarded recursion, and prove properties of our programs using Löb induction.

Streams. The type of guarded recursive streams in GCTT, as with GDTT, are defined as fixed points on the universe:

$$\mathsf{Str}_A\ \triangleq\ \mathsf{fix}\, x.A \times \triangleright[y \leftarrow x].y$$

Note the use of a delayed substitution to transform a term of type $\triangleright\,\mathsf{U}$ to one of type U, as discussed at the start of Sec. 2.2. Desugaring to restate this in terms of dfix, we have

$$\mathsf{Str}_A\ =\ A \times \triangleright[y \leftarrow \mathsf{dfix}^0\, x.A \times \triangleright[y \leftarrow x].y].y$$

The head function $\mathsf{hd} : \mathsf{Str}_A \to A$ is the first projection. The tail function, however, cannot be the second projection, since this yields a term of type

$$\triangleright\left[y \leftarrow \mathsf{dfix}^0\, x.A \times \triangleright\left[y \leftarrow x\right].y\right].y \tag{3}$$

rather than the desired $\triangleright\,\mathsf{Str}_A$. However we are not far off; $\triangleright\,\mathsf{Str}_A$ is judgementally equal to $\triangleright\left[y \leftarrow \mathsf{dfix}^1\, x.A \times \triangleright\left[y \leftarrow x\right].y\right].y$, which is the same term as (3), apart from endpoint 1 replacing 0. The canonical unfold lemma (Lem. 2) tells us that we can build a path in U from Str_A to $A \times \triangleright\,\mathsf{Str}_A$; call this path $\langle i \rangle\,\mathsf{Str}_A^i$. Then we can transport between these types:

$$\mathsf{unfold}\ s \triangleq \mathsf{transp}^i\ \mathsf{Str}_A^i\ s \qquad\qquad \mathsf{fold}\ s \triangleq \mathsf{transp}^i\ \mathsf{Str}_A^{1-i}\ s$$

Note that the compositions of these two operations are path equal to identity functions, but not judgementally equal. We can now obtain the desired tail function $\mathsf{tl} : \mathsf{Str}_A \to \triangleright\,\mathsf{Str}_A$ by composing the second projection with unfold, so $\mathsf{tl}\ s \triangleq (\mathsf{unfold}\ s).2$. Similarly we can define the stream constructor cons (written infix as ::) by using fold:

$$\mathsf{cons} \triangleq \lambda a, s.\, \mathsf{fold}\,(a, s)\ :\ A \to \triangleright\,\mathsf{Str}_A \to \mathsf{Str}_A\,.$$

We now turn to higher order functions on streams. We define $\mathsf{zipWith} : (A \to B \to C) \to \mathsf{Str}_A \to \mathsf{Str}_B \to \mathsf{Str}_C$, the stream function which maps a binary function on two input streams to produce an output stream, as

$$\mathsf{zipWith}\ f \triangleq \mathsf{fix}\, z.\lambda s_1, s_2.f\,(\mathsf{hd}\ s_1)\,(\mathsf{hd}\ s_2)\ ::\ \mathsf{next}\left[\begin{array}{l} z' \leftarrow z \\ t_1 \leftarrow \mathsf{tl}\ s_1 \\ t_2 \leftarrow \mathsf{tl}\ s_2 \end{array}\right].\, z'\, t_1\, t_2.$$

Of course zipWith is definable even with simple types and $\triangleright$, but in GCTT we can go further and prove properties about the function:

▶ **Proposition 4** (zipWith preserves commutativity). *If $f : A \to A \to B$ is commutative, then* $\mathsf{zipWith}\ f : \mathsf{Str}_A \to \mathsf{Str}_A \to \mathsf{Str}_B$ *is commutative.*

Proof. Let $\mathsf{c} : (a_1 : A) \to (a_2 : A) \to \mathsf{Path}\, B\ (f\, a_1\, a_2)\ (f\, a_2\, a_1)$ witness commutativity of f. We proceed by Löb induction, i.e., by assuming

$$\mathsf{ih} : \triangleright\, ((s_1 : \mathsf{Str}_A) \to (s_2 : \mathsf{Str}_A) \to \mathsf{Path}\, B\ (\mathsf{zipWith}\ f\, s_1\, s_2)\ (\mathsf{zipWith}\ f\, s_2\, s_1))\,.$$

Let $i : \mathbb{I}$ be a fresh name, and $s_1, s_2 : \mathsf{Str}_A$. Our aim is to construct a stream v which is $\mathsf{zipWith}\ f\, s_1\, s_2$ when substituting 0 for i, and $\mathsf{zipWith}\ f\, s_2\, s_1$ when substituting 1 for i. An initial attempt at this proof is the term

$$v \triangleq \mathsf{c}\,(\mathsf{hd}\ s_1)\,(\mathsf{hd}\ s_2)\,i\ ::\ \mathsf{next}\left[\begin{array}{l} q \leftarrow \mathsf{ih} \\ t_1 \leftarrow \mathsf{tl}\ s_1 \\ t_2 \leftarrow \mathsf{tl}\ s_2 \end{array}\right].\, q\, t_1\, t_2\, i\ :\ \mathsf{Str}_B,$$

which is equal to

$$f\,(\mathsf{hd}\ s_1)\,(\mathsf{hd}\ s_2)\ ::\ \mathsf{next}\left[\begin{array}{l} t_1 \leftarrow \mathsf{tl}\ s_1 \\ t_2 \leftarrow \mathsf{tl}\ s_2 \end{array}\right].\ \mathsf{zipWith}\ f\, t_1\, t_2$$

when substituting 0 for i, which is $\mathsf{zipWith}\ f\, s_1\, s_2$, but *unfolded once*. Similarly, $v[1/i]$ is $\mathsf{zipWith}\ f\, s_2\, s_1$ unfolded once. Let $\langle j \rangle\, \mathsf{zipWith}^j$ be the canonical unfold lemma associated with zipWith (see Lem. 2). We can now finish the proof by composing v with (the inverse of) the canonical unfold lemma. Diagrammatically, with i along the horizontal axis and j along the vertical:

$$\begin{array}{ccc} \mathsf{zipWith}\ f\, s_1\, s_2 & \dashrightarrow & \mathsf{zipWith}\ f\, s_2\, s_1 \\ \Big\uparrow {\scriptstyle \mathsf{zipWith}^{1-j}\ f\, s_1\, s_2} & & \Big\uparrow {\scriptstyle \mathsf{zipWith}^{1-j}\ f\, s_2\, s_1} \\ f\,(\mathsf{hd}\ s_1)\,(\mathsf{hd}\ s_2)\ :: \\ \mathsf{next}\left[\begin{array}{l} t_1 \leftarrow \mathsf{tl}\ s_1 \\ t_2 \leftarrow \mathsf{tl}\ s_2 \end{array}\right].\ \mathsf{zipWith}\ f\, t_1\, t_2 & \xrightarrow[v]{} & f\,(\mathsf{hd}\ s_2)\,(\mathsf{hd}\ s_1)\ :: \\ & & \mathsf{next}\left[\begin{array}{l} t_2 \leftarrow \mathsf{tl}\ s_2 \\ t_1 \leftarrow \mathsf{tl}\ s_1 \end{array}\right].\ \mathsf{zipWith}\ f\, t_2\, t_1 \end{array}$$

The complete proof term, in the language of the type checker, can be found in Appendix A. ◀

Guarded recursive types with negative variance. A key feature of guarded recursive types are that they support *negative* occurrences of recursion variables. This is important for applications to models of program logics [6]. Here we consider a simple example of a negative variance recursive type, namely $\mathsf{Rec}_A \triangleq \mathsf{fix}\, x.(\triangleright[x' \leftarrow x].x') \to A$, which is path equal to $\triangleright\, \mathsf{Rec}_A \to A$. As a simple demonstration of the expressiveness we gain from negative guarded recursive types, we define a guarded variant of Curry's Y combinator:

$$\begin{array}{lcll} \Delta & \triangleq & \lambda x.f(\mathsf{next}[x' \leftarrow x].((\mathsf{unfold}\, x')x)) & : \quad \triangleright\, \mathsf{Rec}_A \to A \\ \mathsf{Y} & \triangleq & \lambda f.\Delta(\mathsf{next}\,\mathsf{fold}\,\Delta) & : \quad (\triangleright A \to A) \to A, \end{array}$$

where fold and unfold are the transports along the path between Rec_A and $\triangleright\, \mathsf{Rec}_A \to A$. As with zipWith, Y can be defined with simple types and $\triangleright$ [1]; what is new to GCTT is that we can also prove properties about it:

▶ **Proposition 5** (Y is a guarded fixed-point combinator). *$\mathsf{Y}\, f$ is path equal to $f\,(\mathsf{next}(\mathsf{Y}\, f))$, for any $f : \triangleright A \to A$. Therefore, by Prop. 3,* Y *is path equal to* fix.

Proof. $\mathsf{Y}\, f$ simplifies to $f\,(\mathsf{next}(\mathsf{unfold}\,(\mathsf{fold}\,\Delta)\,(\mathsf{next}\,\mathsf{fold}\,\Delta)))$, and $\mathsf{unfold}\,(\mathsf{fold}\,\Delta)$ is path equal to Δ. A congruence over this path yields our path between $\mathsf{Y}\, f$ and $f(\mathsf{next}(\mathsf{Y}\, f))$. ◀

3 Semantics

In this section we sketch the semantics of GCTT. The semantics is based on the category $\widehat{\mathcal{C} \times \omega}$ of presheaves on the category $\mathcal{C} \times \omega$, where $\mathcal{C}$ is the *category of cubes* [10] and ω is the poset of natural numbers. The category of cubes is the opposite of the Kleisli category of the free De Morgan algebra monad on finite sets. More concretely, given a countably infinite set of names $i, j, k, \ldots$, $\mathcal{C}$ has as objects finite sets of names I, J. A morphism $I \to J \in \mathcal{C}$ is a *function* $J \to \mathbf{DM}\,(I)$, where $\mathbf{DM}\,(I)$ is the free De Morgan algebra with generators I.

Following the approach of Cohen et al. [10], contexts of GCTT will be interpreted as objects of $\widehat{\mathcal{C} \times \omega}$. Types in context Γ will be interpreted as pairs (A, c_A) of a presheaf A on the category of elements of Γ and a *composition structure* c_A. We call such a pair a *fibrant* type.

To aid in defining what a composition structure is, and in showing that composition structure is preserved by all the necessary type constructions, we will make use of the internal language of $\widehat{\mathcal{C} \times \omega}$ in the form of *dependent predicate logic*; see for example Phoa [24, App. I].

A type of GCTT in context Γ will then be interpreted as a pair of a type $\Gamma \vdash A$ in the internal language of $\widehat{\mathcal{C} \times \omega}$, and a composition structure c_A, where c_A is a term in the internal language of a specific type $\Phi(\Gamma; A)$, which we define below after introducing the necessary constructs. Terms of GCTT will be interpreted as terms of the internal language. We use *categories with families* [12] as our notion of a model. Due to space limits we omit the precise definition of the category with families here, and refer to the online technical appendix.

The semantics is split into several parts, which provide semantics at different levels of generality.

1. We first show that every presheaf topos with a non-trivial internal De Morgan algebra $\mathbb{I}$ satisfying the disjunction property can be used to give semantics to the subset of the cubical type theory CTT without glueing and the universe. We further show that, for any category $\mathbb{D}$, the category of presheaves on $\mathcal{C} \times \mathbb{D}$ has an interval $\mathbb{I}$, which is the inclusion of the interval in presheaves over the category of cubes $\mathcal{C}$.
2. We then extend the semantics to include glueing and universes. We show that the topos of presheaves $\mathcal{C} \times \mathbb{D}$ for any category $\mathbb{D}$ with an initial object can be used to give semantics to the entire cubical type theory.
3. Finally, we show that the category of presheaves on $\mathcal{C} \times \omega$ gives semantics to delayed substitutions and fixed points. Using these and some additional properties of the delayed substitutions we show in the internal language of $\widehat{\mathcal{C} \times \omega}$ that $\triangleright \xi.A$ has composition whenever A has composition.

Combining all three, we give semantics to GCTT in $\widehat{\mathcal{C} \times \omega}$.

3.1 Model of CTT Without Glueing and the Universe

Let $\mathcal{E}$ be a topos with a natural numbers object, and let $\mathbb{I}$ be a De Morgan algebra internal to $\mathcal{E}$ which satisfies the *finitary disjunction property*, i.e.,

$$(i \vee j) = 1 \implies (i = 1) \vee (j = 1), \quad \text{and} \quad \neg(0 = 1).$$

Faces. Using the interval $\mathbb{I}$ we define the type $\mathbb{F}$ as the image of the function $\cdot = 1 : \mathbb{I} \to \Omega$, where Ω is the subobject classifier. More precisely, $\mathbb{F}$ is the subset type

$$\mathbb{F} \triangleq \{p : \Omega \mid \exists (i : \mathbb{I}), p = (i = 1)\}$$

We will implicitly use the inclusion $\mathbb{F} \to \Omega$. The following lemma states in particular that the inclusion is compatible with all the lattice operations, so omitting it is justified. The disjunction property is crucial for validity of this lemma.

▶ **Lemma 6.**

- $\mathbb{F}$ *is a lattice for operations inherited from* Ω.
- *The corestriction* $\cdot = 1 : \mathbb{I} \to \mathbb{F}$ *is a lattice homomorphism. It is not injective in general.*

Given $\Gamma \vdash \varphi : \mathbb{F}$, we write $[\varphi] \triangleq \mathrm{Id}_{\mathbb{F}}(\varphi, \top)$. Given $\Gamma \vdash A$ and $\Gamma \vdash \varphi : \mathbb{F}$ a *partial element* of type A of *extent* φ is a term t of type $\Gamma \vdash t : \Pi(p : [\varphi]).A$. If we are in a context with $p : [\varphi]$, then we will treat such a partial element t as a term of type A, leaving implicit the application to the proof p, i.e., we will treat t as $t\,p$. We will often write $\Gamma, [\varphi]$ instead of $\Gamma, p : [\varphi]$ when we do not mention the proof term p explicitly in the rest of the judgement. This is justified since inhabitants of $[\varphi]$ are unique up to judgemental equality (recall that dependent predicate logic is a logic over an extensional dependent type theory). Given $\Gamma, p : [\varphi] \vdash B$ we write B^{φ} for the dependent function space $\Pi(p : [\varphi]).B$ and again leave the proof p implicit.

For a term $\Gamma, p : [\varphi] \vdash u : A$ we define $A[\varphi \mapsto u] \triangleq \Sigma(a : A).\,(\mathrm{Id}_A(a, u))^{\varphi}$.

Compositions. Faces allow us to define the type of *compositions* $\Phi(\Gamma; A)$. Homotopically, compositions allow us to put a lid on a box [10]. Given $\Gamma \vdash A$ we define the corresponding type of compositions as

$$\Phi(\Gamma; A) \triangleq \Pi(\gamma : \mathbb{I} \to \Gamma)(\varphi : \mathbb{F})\,(u : \Pi(i : \mathbb{I}).\,(A(\gamma(i)))^{\varphi})\,. \\ A(\gamma(0))[\varphi \mapsto u(0)] \to A(\gamma(1))[\varphi \mapsto u(1)].$$

Here we treat the context Γ as a closed type. This is justified because there is a canonical bijection between contexts and closed types of the internal language. The notation $A(\gamma(i))$ means substitution along the (uncurried) γ.

Due to lack of space we do not show how the standard constructs of the type theory are interpreted. We only sketch how the following composition term is interpreted in terms of the composition in the model.

$$\frac{\Gamma \vdash \varphi : \mathbb{F} \qquad \Gamma, i : \mathbb{I} \vdash A \qquad \Gamma, \varphi, i : \mathbb{I} \vdash u : A \qquad \Gamma \vdash a_0 : A[0/i][\varphi \mapsto u[0/i]]}{\Gamma \vdash \mathsf{comp}^i\ A\ [\varphi \mapsto u]\ a_0 : A[1/i][\varphi \mapsto u[1/i]]}.$$

By assumption we have c_A of type $\Phi(\Gamma, i : \mathbb{I}; A)$ and u and a_0 are interpreted as terms in the internal language of the corresponding types. The interpretation of composition is the term

$$\gamma : \Gamma \vdash c_A\,(\lambda(i : \mathbb{I}).(\gamma, i))\,\varphi\,(\lambda(i : \mathbb{I})(p : [\varphi]).u)\,a_0 : A(\gamma(1))[\varphi \mapsto u(1)]$$

where we have omitted writing the proof $u(0) = a_0$ on $[\varphi]$.

Concrete models. The category of cubical sets has an internal interval type satisfying the disjunction property [10]. It is the functor mapping $I \in \mathcal{C}$ to $\mathbf{DM}\,(I)$. Since the theory of a De Morgan algebra with $0 \neq 1$ and the disjunction property is geometric [16, Section X.3] we have that for any topos $\mathcal{F}$ and geometric morphism $\varphi : \mathcal{F} \to \widehat{\mathcal{C}}$, $\varphi^*(\mathbb{I}) \in \mathcal{F}$ is a De Morgan algebra with the disjunction property[6]. In particular, given any category $\mathbb{D}$ there is a projection functor $\pi : \mathcal{C} \times \mathbb{D} \to \mathcal{C}$ which induces the (essential) geometric morphism $\pi^* \dashv \pi_* : \widehat{\mathcal{C} \times \mathbb{D}} \to \widehat{\mathcal{C}}$, where π^* is precomposition with π, and π_* takes limits along $\mathbb{D}$.

[6] A statement very close to this can be used as a characterisation of $\widehat{\mathcal{C}}$: this topos classifies the geometric theory of flat De Morgan algebras [25].

Summary. With the semantic structures developed thus far we can give semantics to the subset of CTT without glueing and the universe.

3.2 Adding Glueing and the Universe

The glueing construction [10, Sec. 6] is used to prove both fibrancy and, subsequently, univalence of the universe of fibrant types. Concretely, given

$$\Gamma \vdash \varphi : \mathbb{I} \qquad \Gamma, [\varphi] \vdash T \qquad \Gamma \vdash A \qquad \Gamma \vdash w : (T \to A)^{\varphi}$$

we define the type Glue $[\varphi \mapsto (T, w)]\ A$ in two steps. First we define the type[7]

$$Glue'_{\Gamma}(\varphi, T, A, w) \triangleq \sum_{a:A} \sum_{t:T^{\varphi}} \prod_{p:[\varphi]} wp(tp) = a.$$

For this type we have the following property $\Gamma, [\varphi] \vdash T \cong Glue'_{\Gamma}(\varphi, T, A, w)$. However, we need an equality, not an isomorphism, to obtain the correct typing rules. The technical appendix provides a general strictification lemma which allows us to define the type *Glue*.

To show that the type Glue $[\varphi \mapsto (T, w)]\ A$ is fibrant we need to additionally assume that the map $\varphi \mapsto \lambda_.\varphi : \mathbb{F} \to (\mathbb{I} \to \mathbb{F})$ has an internal right adjoint $\forall$. Such a right adjoint exists in all toposes $\widehat{\mathcal{C} \times \mathbb{D}}$, for any small category $\mathbb{D}$ with an initial object.

Universe of fibrant types. Given a (Grothendieck) universe $\mathfrak{U}$ in the meta-theory, the Hofmann-Streicher universe [14] $\mathcal{U}^{\omega}$ in $\widehat{\mathcal{C} \times \omega}$ maps (I, n) to the set of functors valued in $\mathfrak{U}$ on the category of elements of $y(I, n)$, where y is the Yoneda embedding. As in Cohen et al. [10] we define the universe of *fibrant* types $\mathcal{U}^{\omega}_f$ by setting $\mathcal{U}^{\omega}_f(I, n)$ to be the set of fibrant types in context $y(I, n)$. The universe $\mathcal{U}^{\omega}_f$ satisfies the rules

$$\frac{\Gamma \vdash a : \mathcal{U} \qquad \vdash \mathbf{c} : \Phi(\Gamma; \mathrm{El}(a))}{\Gamma \vdash (\!|a, \mathbf{c}|\!) : \mathcal{U}_f} \qquad \frac{\Gamma \vdash a : \mathcal{U}_f}{\Gamma \vdash \mathrm{El}(a)} \qquad \frac{\Gamma \vdash a : \mathcal{U}_f}{\vdash \mathrm{Comp}(a) : \Phi(\Gamma; \mathrm{El}(a))}$$

Using the glueing operation, one shows that the universe of fibrant types is *itself* fibrant and, moreover, that it is univalent.

3.3 Adding the Later Type-Former

We now fix the site to be $\mathcal{C} \times \omega$. From the previous sections we know that $\widehat{\mathcal{C} \times \omega}$ gives semantics to CTT. The new constructs of GDTT are the $\triangleright$ type-former and its delayed substitutions, and guarded fixed points. Continuing to work in the internal language, we first show that the internal language of $\widehat{\mathcal{C} \times \omega}$ can be extended with these constructions, allowing interpretation of the subset of the type theory GDTT without clock quantification [8]. Due to lack of space we omit the details of this part, but do remark that $\triangleright$ is defined as

$$(\triangleright(X))(I, n) \begin{cases} \{\star\} & \text{if } n = 0 \\ X(I, m) & \text{if } n = m + 1 \end{cases}$$

The essence of this definition is that $\triangleright$ depends only on the "ω component" and ignores the "$\mathcal{C}$ component". Verification that all the rules of GDTT are satisfied is therefore very similar to the verification that the topos $\widehat{\omega}$ is a model of the same subset of GDTT.

[7] This type is already present in Kapulkin at al. [15, Thm 3.4.1].

The only additional property we need now is that $\triangleright$ preserves compositions, in the sense that if we have a delayed substitution $\vdash \xi : \Gamma \twoheadrightarrow \Gamma'$ and a type $\Gamma, \Gamma' \vdash A$ together with a closed term $\mathbf{c}_A$ of type $\Phi(\Gamma, \Gamma'; A)$ then we can construct $\mathbf{c}'_{\triangleright\xi.A}$ of type $\Phi(\Gamma; \triangleright\xi.A)$.

The following lemma uses the notion of a type $\Gamma \vdash A$ being *constant with respect to* ω. This notion is a natural generalisation to types-in-context of the property that a presheaf is in the image of the functor π^*. We refer to the online technical appendix for the precise definition. Here we only remark that the interval type $\mathbb{I}$ is constant with respect to ω, as is the type $\Gamma \vdash [\varphi]$ for any term $\Gamma \vdash \varphi : \mathbb{F}$.

▶ **Lemma 7.** *Assume* $\Gamma \vdash A$, $\Gamma, \Gamma', x : A \vdash B$ *and* $\vdash \xi : \Gamma \twoheadrightarrow \Gamma'$, *and further that* A *is constant with respect to* ω. *Then the following two types are isomorphic*

$$\Gamma \vdash \triangleright\xi.\Pi(x : A).B \cong \Pi(x : A).\triangleright\xi.B \tag{4}$$

and the canonical morphism $\lambda f.\lambda x.\, \mathsf{next}\,[\xi, f' \leftarrow f]\,.\, f'\,x$ *from left to right is an isomorphism.*

▶ **Corollary 8.** *If* $\Gamma \vdash \varphi : \mathbb{F}$ *then we have an isomorphism of types*

$$\Gamma \vdash \triangleright\xi.\Pi(p : [\varphi]).B \cong \Pi(x : [\varphi]).\triangleright\xi.B. \tag{5}$$

▶ **Lemma 9** ($\triangleright\xi$-types preserve compositions)**.** *If* $\triangleright\xi.A$ *is a well-formed type in context* Γ *and we have a composition term* $\mathbf{c}_A : \Phi(\Gamma, \Gamma'; A)$, *then there is a composition term* $\mathbf{c} : \Phi(\Gamma; \triangleright\xi.A)$.

Proof. We show the special case with an empty delayed substitution. For the more general proof we refer to the technical appendix. Assume we have a composition $\mathbf{c}_A : \Phi(\Gamma; A)$. Our goal is to find a term $\mathbf{c} : \Phi(\Gamma; \triangleright A)$, so we first introduce some variables:

$$\gamma : \mathbb{I} \to \Gamma \qquad \varphi : \mathbb{F} \qquad u : \Pi(i : \mathbb{I}).\,((\triangleright A)(\gamma\, i))^{\varphi} \qquad a_0 : (\triangleright A)(\gamma\, 0)[\varphi \mapsto u\, 0].$$

Using the isomorphisms from Cor. 8 and Lem. 7 we obtain a term $\tilde{u} : \triangleright(\Pi(i : \mathbb{I}).(A(\gamma\, i))^{\varphi})$ isomorphic to u. We can now – almost – write the term

$$\mathsf{next}\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].\, \mathbf{c}_A\, \gamma\, \varphi\, u'\, a'_0 \;:\; \triangleright(A(\gamma\, 1)), \tag{$*$}$$

what is missing is to check that $a'_0 = u'\, 0$ on the extent φ, so that we can legally apply $\mathbf{c}_A$; this is equivalent to saying that the type $\triangleright\,[u' \leftarrow \tilde{u}, a'_0 \leftarrow a_0]\,.\,\mathrm{Id}_{A(\gamma\, 0)}(a'_0, u'\, 0)^{\varphi}$ is inhabited. We transform this type as follows:

$$\begin{aligned} \triangleright\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].\mathrm{Id}(a'_0, u'\, 0)^{\varphi} &\cong \left(\triangleright\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].\mathrm{Id}(a'_0, u'\, 0)\right)^{\varphi} && \text{(Cor. 8)} \\ &= \left(\mathrm{Id}(\mathsf{next}\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].a'_0, \mathsf{next}\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].u'\, 0)\right)^{\varphi} \\ &= (\mathrm{Id}(a_0, u\, 0))^{\varphi}, \end{aligned}$$

where the last equality uses that $\tilde{u}$ is defined using the inverse of $\lambda f \lambda x.\, \mathsf{next}\,[f' \leftarrow f]\,.\, f'\,x$ (Lem. 7). By assumption it is the case that $(\mathrm{Id}(a_0, u\, 0))^{\varphi}$ is inhabited, and therefore $(*)$ is well-defined. It remains only to check that $(*)$ is equal to $u\, 1$ on the extent φ, but this follows from the equalities of $\mathbf{c}_A$ and by the definition of $\tilde{u}$ (Lem. 7). Assuming φ, we have

$$\mathsf{next}\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].\, \mathbf{c}_A\, \gamma\, \varphi\, u'\, a'_0 = \mathsf{next}\left[\begin{array}{l} u' \leftarrow \tilde{u} \\ a'_0 \leftarrow a_0 \end{array}\right].\, u'\, 1 = u\, 1. \qquad \blacktriangleleft$$

Summary. In this section we have highlighted the key ingredients that go into a sound interpretation of GCTT in $\widehat{\mathcal{C} \times \omega}$. For the precise statement of the interpretation of all the constructs, and the soundness theorem, we refer to the online technical appendix.

4 Conclusion

In this paper we have made the following contributions:

- We introduce guarded cubical type theory (GCTT), which combines features of cubical type theory (CTT) and guarded dependent type theory (GDTT). The path equality of CTT is shown to support reasoning about extensional properties of guarded recursive operations, and we use the interval of CTT to constrain the unfolding of fixed-points.
- We show that CTT can be modelled in any presheaf topos with an internal non-trivial De Morgan algebra with the disjunction property, an operator $\forall$, and a universe of fibrant types. Most of these constructions are done via the internal logic. We then show that a class of presheaf models of the form $\widehat{\mathcal{C} \times \mathbb{D}}$, for any category $\mathbb{D}$ with an initial object, satisfy the above axioms and hence gives rise to a model of CTT.
- We give semantics to GCTT in the topos of presheaves over $\mathcal{C} \times \omega$.

Further work. We wish to establish key *syntactic properties* of GCTT, namely decidable type-checking and canonicity for base types. Our prototype implementation establishes some confidence in these properties.

We wish to further extend GCTT with *clock quantification* [3], such as is present in GDTT. Clock quantification allows for the controlled elimination of the later type-former, and hence the encoding of first-class coinductive types via guarded recursive types. The generality of our approach to semantics in this paper should allow us to build a model by combining cubical sets with the presheaf model of GDTT with multiple clocks [7]. The main challenges lie in ensuring decidable type checking (GDTT relies on certain rules involving clock quantifiers which seem difficult to implement), and solving the *coherence problem* for clock substitution.

Finally, some higher inductive types, like the truncation, can be added to CTT. We would like to understand how these interact with $\triangleright$.

Related work. Another type theory with a computational interpretation of functional extensionality, but without equality reflection, is observational type theory (OTT) [2]. We found CTT's prototype implementation, its presheaf semantics, and its interval as a tool for controlling unfoldings, most convenient for developing our combination with GDTT, but extending OTT similarly would provide an interesting comparison.

Spitters [25] used the interval of the internal logic of cubical sets to model identity types. Coquand [11] defined the composition operation internally to obtain a model of type theory. We have extended both these ideas to a full model of CTT. Recent independent work by Orton and Pitts [23] axiomatises a model for CTT without a universe, again building on Coquand [11]. With the exception of the absence of the universe, their development is more general than ours. Our semantic developments are sufficiently general to support the sound addition of guarded recursive types to CTT.

Acknowledgements. We gratefully acknowledge our discussions with Thierry Coquand, and the comments of our reviewers. This research was supported in part by the ModuRes Sapere Aude Advanced Grant from The Danish Council for Independent Research for the Natural Sciences (FNU). Aleš Bizjak was supported in part by a Microsoft Research PhD grant.

References

1 Andreas Abel and Andrea Vezzosi. A formalized proof of strong normalization for guarded recursive types. In *APLAS*, pages 140–158, 2014.

2 Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *PLPV*, pages 57–68, 2007.

3 Robert Atkey and Conor McBride. Productive coprogramming with guarded recursion. In *ICFP*, pages 197–208, 2013.

4 Lars Birkedal and Rasmus Ejlers Møgelberg. Intensional type theory with guarded recursive types qua fixed points on universes. In *LICS*, pages 213–222, 2013.

5 Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *LMCS*, 8(4), 2012. doi:10.2168/LMCS-8(4:1)2012.

6 Lars Birkedal, Bernhard Reus, Jan Schwinghammer, Kristian Støvring, Jacob Thamsborg, and Hongseok Yang. Step-indexed Kripke models over recursive worlds. In *POPL*, pages 119–132, 2011.

7 Aleš Bizjak and Rasmus Ejlers Møgelberg. A model of guarded recursion with clock synchronisation. In *MFPS*, pages 83–101, 2015.

8 Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. Guarded dependent type theory with coinductive types. In *FoSSaCS*, pages 20–35, 2016.

9 Ranald Clouston, Aleš Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. Programming and reasoning with guarded recursion for coinductive types. In *FoSSaCS*, pages 407–421, 2015.

10 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. Unpublished, 2016.

11 Thierry Coquand. Internal version of the uniform Kan filling condition. Unpublished, 2015. URL: http://www.cse.chalmers.se/~coquand/shape.pdf.

12 Peter Dybjer. Internal type theory. In *TYPES'95*, pages 120–134, 1996.

13 Martin Hofmann. *Extensional constructs in intensional type theory.* Springer, 1997.

14 Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes. Unpublished, 1999. URL: http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf.

15 Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. The simplicial model of univalent foundations. *arXiv:1211.2851*, 2012.

16 Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic.* Springer, 1992. doi:10.1007/978-1-4612-0927-0.

17 Per Martin-Löf. An intuitionistic theory of types: predicative part. In *Logic Colloquium 1973*, pages 73–118, 1975.

18 The Coq development team. *The Coq proof assistant reference manual.* LogiCal Project, 2004. Version 8.0. URL: http://coq.inria.fr.

19 Conor McBride and Ross Paterson. Applicative programming with effects. *J. Funct. Programming*, 18(1):1–13, 2008.

20 Rasmus Ejlers Møgelberg. A type theory for productive coprogramming via guarded recursion. In *CSL-LICS*, 2014.

21 Hiroshi Nakano. A modality for recursion. In *LICS*, pages 255–266, 2000.

22 Ulf Norell. *Towards a practical programming language based on dependent type theory.* PhD thesis, Chalmers University of Technology, 2007.

23 Ian Orton and Andrew M. Pitts. Axioms for modelling cubical type theory in a topos. In *CSL*, 2016.

24 Wesley Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. Technical Report ECS-LFCS-92-208, LFCS, University of Edinburgh, 1992.

25 Bas Spitters. Cubical sets as a classifying topos. *TYPES*, 2015.
26 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations for Mathematics*. Institute for Advanced Study, 2013. URL: `http://homotopytypetheory.org/book`.

A zipWith Preserves Commutativity

We provide a formalisation of Sec. 2.4 which can be verified by our type checker[8].

```
module zipWith_preserves_comm where

Id (A : U) (a0 a1 : A) : U = IdP (⟨i⟩ A) a0 a1
data nat = Z | S (n : nat)

-- Streams of natural numbers
StrF (S : ▷ U) : U = (n : nat) * ▷ [S' ← S] S'

Str : U = fix (StrF Str)

-- The canonical unfold lemma for Str
StrUnfoldPath : Id U Str (StrF (next Str))
 = ⟨i⟩ StrF (dfix U StrF [(i=1)])

unfoldStr (s : Str) : (n : nat) * ▷ Str
 = transport StrUnfoldPath s

foldStr (s : (n : nat) * ▷ Str) : Str
 = transport (⟨i⟩ StrUnfoldPath @ -i) s

cons (n : nat) (s : ▷ Str) : Str = foldStr (n, s)
head (s : Str) : nat = s.1
tail (s : Str) : ▷ Str = (unfoldStr s).2

-- Defining zipWith
zipWithF (f : nat → nat → nat) (rec : ▷ (Str → Str → Str))
 : Str → Str → Str
 = (λ (s1 s2 : Str) →
      (cons (f (head s1) (head s2))
            (next [zipWith' ← rec, s1' ← tail s1 , s2' ← tail s2]
                  zipWith' s1' s2')))

zipWith (f : nat → nat → nat) : Str → Str → Str
 = fix (zipWithF f zipWith)

zipWithUnfoldPath (f : nat → nat → nat)
 : Id (Str → Str → Str)
      (zipWith f)
      (zipWithF f (next (zipWith f)))
 = ⟨i⟩ zipWithF f (dfix (Str → Str → Str) (zipWithF f) [(i=1)])

-- Commutativity property
comm (f : nat → nat → nat) : U = (m n : nat) → Id nat (f m n) (f n m)

-- zipWith preserves commutativity.
zipWith_preserves_comm (f : nat → nat → nat) (c : comm f)
 : (s1 s2 : Str) → Id Str (zipWith f s1 s2) (zipWith f s2 s1)
 = fix
   (λ (s1 s2 : Str) →
      ⟨i⟩ comp (⟨_⟩ Str)
               (cons (c (head s1) (head s2) @ i)
                     (next [q ← zipWith_preserves_comm
                           ,t1 ← tail s1
                           ,t2 ← tail s2]
                           q t1 t2 @ i))
               [(i=0) → ⟨j⟩ zipWithUnfoldPath f @ -j s1 s2
               ,(i=1) → ⟨j⟩ zipWithUnfoldPath f @ -j s2 s1])
```

[8] This file, among other examples, is available in the `gctt-examples` folder in the type-checker repository.

Axioms for Modelling Cubical Type Theory in a Topos

Ian Orton[1] **and Andrew M. Pitts**[2]

1 **University of Cambridge Computer Laboratory, Cambridge, U.K.**
`Ian.Orton@cl.cam.ac.uk`
2 **University of Cambridge Computer Laboratory, Cambridge, U.K.**
`Andrew.Pitts@cl.cam.ac.uk`

Abstract

The homotopical approach to intensional type theory views proofs of equality as paths. We explore what is required of an interval-like object $\mathtt{I}$ in a topos to give a model of type theory in which elements of identity types are functions with domain $\mathtt{I}$. Cohen, Coquand, Huber and Mörtberg give such a model using a particular category of presheaves. We investigate the extent to which their model construction can be expressed in the internal type theory of any topos and identify a collection of quite weak axioms for this purpose. This clarifies the definition and properties of the notion of uniform Kan filling that lies at the heart of their constructive interpretation of Voevodsky's univalence axiom. Furthermore, since our axioms can be satisfied in a number of different ways, we show that there is a range of topos-theoretic models of homotopy type theory in this style.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases models of dependent type theory, homotopy type theory, cubical sets, cubical type theory, topos, univalence

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.24

1 Introduction

Cubical type theory [11] provides a constructive justification of Voevodsky's univalence axiom, an axiom that has important consequences for the formalisation of mathematics within Martin-Löf type theory [33]. Working informally in constructive set theory, Cohen *et al* [11] give a model of their type theory using the category $\hat{\mathcal{C}}$ of set-valued contravariant functors on a small category $\mathcal{C}$ which is the Lawvere theory for de Morgan algebra [5, Chapter XI]; see [31]. The representable functor on the generic de Morgan algebra in $\mathcal{C}$ is used as an interval object $\mathtt{I}$ in $\hat{\mathcal{C}}$, with proofs of equality modelled by the corresponding notion of path, that is, by morphisms with domain $\mathtt{I}$. Cohen *et al* call the objects of $\hat{\mathcal{C}}$ *cubical sets*. They have a richer structure compared with previous, synonymous notions [7, 21]. For one thing they allow path types to be modelled simply by exponentials $X^{\mathtt{I}}$, rather than by name abstractions [30, Chapter 4]. More importantly, the de Morgan algebra operations endow $\mathtt{I}$ with structure that considerably simplifies the definition and properties of the constructive notion of Kan filling that lies at the heart of [11]. In particular, the filling operation is obtained from a simple special case that *composes* a filling at one end of the interval to a filling at the other end. Coquand [12] has suggested that this distinctive composition operation can be understood in terms of the properties of partial elements and their extension to total elements, within the internal higher-order logic of toposes [24]. In this paper we show that that is indeed the case and usefully so. In particular, the *uniformity* condition on

© Ian Orton and Andrew M. Pitts;
licensed under Creative Commons License CC-BY

The interval $\mathtt{I}$ is connected

$$\mathtt{ax}_1 : [\forall(\varphi : \mathtt{I} \to \Omega).\ (\forall(i : \mathtt{I}).\ \varphi\, i \vee \neg\varphi\, i) \Rightarrow (\forall(i : \mathtt{I}).\ \varphi\, i) \vee (\forall(i : \mathtt{I}).\ \neg\varphi\, i)]$$

has distinct end-points $\mathtt{0}, \mathtt{1} : \mathtt{I}$

$$\mathtt{ax}_2 : [\neg(\mathtt{0} = \mathtt{1})]$$

and has a connection algebra structure $_ \sqcap _, _ \sqcup _ : \mathtt{I} \to \mathtt{I} \to \mathtt{I}$

$$\mathtt{ax}_3 : [\forall(i : \mathtt{I}).\ \mathtt{0} \sqcap x = \mathtt{0} = x \sqcap \mathtt{0} \ \wedge\ \mathtt{1} \sqcap x = x = x \sqcap \mathtt{1}]$$
$$\mathtt{ax}_4 : [\forall(i : \mathtt{I}).\ \mathtt{0} \sqcup x = x = x \sqcup \mathtt{0} \ \wedge\ \mathtt{1} \sqcup x = \mathtt{1} = x \sqcup \mathtt{1}].$$

Cofibrant propositions $\mathtt{Cof} = \{\varphi : \Omega \mid \mathtt{cof}\ \varphi\}$ (where $\mathtt{cof} : \Omega \to \Omega$) include end-point-equality

$$\mathtt{ax}_5 : [\forall(i : \mathtt{I}).\ \mathtt{cof}(i = \mathtt{0}) \ \wedge\ \mathtt{cof}(i = \mathtt{1})]$$

and are closed under binary disjunction

$$\mathtt{ax}_6 : [\forall(\varphi\ \psi : \Omega).\ \mathtt{cof}\ \varphi \Rightarrow \mathtt{cof}\ \psi \Rightarrow \mathtt{cof}(\varphi \vee \psi)]$$

and dependent conjunction

$$\mathtt{ax}_7 : [\forall(\varphi\ \psi : \Omega).\ \mathtt{cof}\ \varphi \Rightarrow (\varphi \Rightarrow \mathtt{cof}\ \psi) \Rightarrow \mathtt{cof}(\varphi \wedge \psi)].$$

Strictness postulate for universe $\mathcal{U}$: any cofibrant-partial type A that is isomorphic to a total type B everywhere that A is defined, can be extended to a total type B' that is isomorphic to B:

$$\mathtt{ax}_8 : \{\varphi : \mathtt{Cof}\}(A : [\varphi] \to \mathcal{U})(B : \mathcal{U})(s : (u : [\varphi]) \to A\,u \cong B) \to$$
$$(B' : \mathcal{U}) \times \{s' : B' \cong B \mid \forall(u : [\varphi]).\ A\,u = B' \wedge s\,u = s'\}.$$

Figure 1 The axioms.

composition operations [11, Definition 13], which allows one to avoid the non-constructive aspects of the classical notion of Kan filling [6], becomes automatic when the operations are formulated internally. Our approach has the usual benefit of axiomatics – helping to clarify exactly which properties of a topos are sufficient to carry out each of the various constructions used to model cubical type theory [11]. For example, we show that something weaker than de Morgan algebra on the interval object $\mathtt{I}$ is sufficient and as a result other models emerge, including one based on constructive simplicial sets.

To accomplish all this, we find it helpful to work not in the higher-order predicate logic of toposes, but in an extensional type theory equipped with an impredicative universe of propositions Ω, standing for the subobject classifier of the topos [27]. Working in such a language, our axiomatisation concerns two structures that a topos $\mathcal{E}$ may possess: an object $\mathtt{I}$ that is endowed with some elementary characteristics of the unit interval; and a subobject of propositions $\mathtt{Cof} \rightarrowtail \Omega$ whose elements we call *cofibrant propositions* and which determine the subobjects that are relevant for a Kan-like notion of filling (for example in the case of [11], subobjects generated by unions of faces of hypercubes). Working internally with cofibrant propositions rather than externally with a class of cofibrant subobjects (monomorphisms) leads to an appealingly simple notion of *fibration* (Section 4), with that of Cohen *et al* as

an instance when the topos is $\hat{\mathcal{C}}$. These fibrations are type-families equipped with extra structure (*composition* operations) which are supposed to model intensional Martin-Löf type theory, when organised as a Category with Families [14], say. In order that they do so, we make a series of postulates about the interval and cofibrant subobjects that are true of the presheaf model in [11]. For ease of reference these axioms are collected together in Figure 1, written in the language described in Section 2. Axiom $\mathtt{ax}_1$ expresses that the interval `I` is internally connected, in the sense that any decidable subset of its elements is either empty or the whole of `I`. It is used at the end of Section 4.2 to show that the topos natural number object is fibrant and that fibrations are closed under binary coproducts; and it also gets used in proving properties of the glueing construct mentioned below. Axioms $\mathtt{ax}_3$ and $\mathtt{ax}_4$ give what we call a *path connection algebra* structure on `I`; they capture some very simple properties of the minimum and maximum operations on the unit interval $[0, 1]$ of real numbers that suffice to ensure contractibility of singleton types (Section 3) and, in combination with $\mathtt{ax}_2$, $\mathtt{ax}_5$ and $\mathtt{ax}_6$, to define path lifting from composition for fibrations (Section 4.2). Indeed only axioms $\mathtt{ax}_2$–$\mathtt{ax}_6$ are needed to show that fibrations provide a model of Π- and Σ-types; and furthermore to show that the path types determined by the interval object `I` (Section 3) satisfy the rules for identity types propositionally [13, 34]. Axiom $\mathtt{ax}_7$ is used to get from these propositional identity types to the proper, definitional identity types of Martin-Löf type theory, via a version of Swan's construction [32]; see Section 4.3. In Section 5 we consider univalence [33, Section 2.10] – the correspondence between type-valued paths in a universe and functions that are equivalences modulo path-based equality. To do so we give a non-strict, "up-to-isomorphism" version of the *glueing* construct of Cohen *et al* in the internal type theory of the topos. Axiom $\mathtt{ax}_8$ allows us to regain the strict form of glueing used in [11]. Our development also differs from that of Cohen *et al* by avoiding the need to postulate that cofibrant propositions are closed under `I`-indexed intersection. However, we do not yet have an internalisation of the modified form of Hofmann-Streicher universe construction [20] used in [11] to obtain a fibrant universe satisfying the univalence axiom; and closure of `Cof` under `I`-indexed intersection may be needed for that.

In Section 6 we indicate why the model in [11] satisfies our axioms and more generally which other presheaf toposes satisfy them. There is some freedom in choosing the subobject of cofibrant propositions. The path-connection algebra structure we assume for the interval `I` (axioms $\mathtt{ax}_3$ and $\mathtt{ax}_4$) is much weaker than being a de Morgan algebra. In particular, we can avoid the use of a de Morgan involution operation and as a result the topos of simplicial sets supports a model of the axioms. It remains to be seen whether the univalent universe construction of [11] transfers to this constructive simplicial set model and how that relates to the classical simplicial model of univalent foundations [23]. In Section 7 we conclude by considering some other related work and future directions.

Agda formalisation

The definitions and constructions we carry out in the internal type theory of toposes are sufficiently involved to warrant machine-assisted formalisation. Our tool of choice is Agda [3]. We persuaded it to provide an impredicative universe of mere propositions [33, Section 3.3] using a method due to Escardo [15]. This gives an intensional, proof-relevant version of the subobject classifier Ω and of the type theory described in Section 2. To this we add postulates corresponding to the axioms in Figure 1. We also made modest use of the facility for user-defined rewriting in recent versions of Agda [10], in order to make the connection algebra axioms $\mathtt{ax}_3$ and $\mathtt{ax}_4$ definitional, rather than just propositional equalities, thereby eliminating a few proofs in favour of computation. Using Agda required us to construct and

pass around many proof details that are omitted or elided in the paper version; we found this to be quite bearable and also invaluable for getting the details right. Our development can be found at `http://www.cl.cam.ac.uk/~rio22/agda/cubical-topos/root.html`.

2 Internal Type Theory of a Topos

We rely on the categorical semantics of dependent type theory in terms of *categories with families* (CwF) [14]. For each topos $\mathcal{E}$ (with subobject classifier $\top : 1 \to \Omega$) one can find a CwF with the same objects, such that the category of families at each object X is equivalent to the slice category $\mathcal{E}/X$. This can be done in a number of different ways; for example [29, Example 6.14], or the more recent references [23, Section 1.3], [26] and [4], which cater for categories more general than a topos (and for contextual/comprehension categories rather than CwFs in the first two cases). Using the objects, families and elements of this CwF as a signature, we get an internal type theory along the lines of those discussed in [27], canonically interpreted in the CwF in the standard fashion [19]. We make definitions and postulates in this internal language for $\mathcal{E}$ using a concrete syntax inspired by Agda [3]. Dependent function types are written as $(x : A) \to B$; their canonical terms are function abstractions, written as $\lambda(x : A) \to t$. Dependent product types are written as $(x : A) \times B$; their canonical terms are pairs, written as (s, t). The subobject classifier Ω becomes an impredicative universe of propositions in the internal type theory with logical connectives, equality and quantifiers $\top, \bot, \neg, \wedge, \vee, \Rightarrow, =, \forall(x : A), \exists(x : A)$. Its universal property gives rise to comprehension subtypes: given $\Gamma, x : A \vdash \varphi(x) : \Omega$, then $\Gamma \vdash \{x : A \mid \varphi(x)\}$ is a type whose terms are those $t : A$ for which $\varphi(t)$ is provable, with the proof being treated irrelevantly.[1] Taking $A = 1$ to be terminal, for each $\varphi : \Omega$ we have a type whose inhabitation corresponds to provability of φ:

$$[\varphi] \triangleq \{_ : 1 \mid \varphi\} \tag{1}$$

We will make extensive use of these types in connection with the partial elements of a type; see Section 4.1.

Instead of quantifying externally over the objects, families and elements of the CwF associated with $\mathcal{E}$, we will assume $\mathcal{E}$ comes with an internal full subtopos $\mathcal{U}$. In the internal language we use $\mathcal{U}$ as a Russell-style universe (that is, if $A : \mathcal{U}$, then A itself denotes a type) containing Ω and closed under forming products, exponentials and comprehension subtypes.

3 Path Types

The homotopical approach to type theory [33] views elements of identity types as paths between the two elements being equated. We try to take this literally, using paths in a topos $\mathcal{E}$ that are morphisms out of a distinguished object $\mathtt{I}$, called the *interval*. We assume it is equipped with morphisms $\mathtt{0}, \mathtt{1} : 1 \to \mathtt{I}$ and $_ \sqcap _, _ \sqcup _ : \mathtt{I} \to \mathtt{I} \to \mathtt{I}$ satisfying axioms $\mathtt{ax_1}$–$\mathtt{ax_4}$ in Figure 1. Axiom $\mathtt{ax_1}$ is an internal connectedness property of the interval that we will not need until Section 6.1. Axiom $\mathtt{ax_2}$ says that the interval is non-trivial. Axioms $\mathtt{ax_3}$ and $\mathtt{ax_4}$ endow it with a form of *path-connection* algebra structure [9] which is used to ensure contractibility of singleton types (see below) and to define path lifting from composition for fibrations (see Section 4.2). In the model of [11] the connection algebra structure is given by

[1] Our Agda development is proof relevant, so that terms of comprehension types contain a proof of membership as a component.

the lattice structure of the interval, taking $_ \sqcap _$ to be binary meet, $_ \sqcup _$ to be binary join and using the fact that $\mathtt{0}$ and $\mathtt{1}$ are respectively least and greatest elements.

▶ Remark 3.1 (de Morgan involution). In the model of [11] $\mathtt{I}$ is not just a lattice, but also has an involution operation $1 - (_) : \mathtt{I} \to \mathtt{I}$ (so that $(1 - (1 - i) = i)$ making $\sqcup$ the de Morgan dual of $\sqcap$, in the sense that $i \sqcup j = 1 - ((1 - i) \sqcap (1 - j))$. We have found that although this involution structure is convenient, it is not strictly necessary for the constructions that follow. Instead we can just give a $\mathtt{0}$-version and a $\mathtt{1}$-version of certain concepts; for example, "composing from $\mathtt{1}$ to $\mathtt{0}$" as well as "composing from $\mathtt{0}$ to $\mathtt{1}$" in Section 4.2.

Given $A : \mathcal{U}$, we call terms of type $\mathtt{I} \to A$ *paths* in A. The *path type* associated with A is $_ \sim _ : A \to A \to \mathcal{U}$ where

$$a_0 \sim a_1 \triangleq \{p : \mathtt{I} \to A \mid p\,\mathtt{0} = a_0 \wedge p\,\mathtt{1} = a_1\} \tag{2}$$

Can these types be used to model the rules for Martin-Löf identity types? We can certainly interpret the identity introduction rule (reflexivity), since degenerate paths given by constant functions

$$\mathtt{k}\,a\,i \triangleq a \tag{3}$$

satisfy $\mathtt{k} : \{A : \mathcal{U}\}(a : A) \to a \sim a$.[2] However, we need further assumptions to interpret the identity elimination rule, otherwise known as path induction [33, Section 1.12.1]. Coquand has given an alternative (propositionally equivalent) formulation of identity elimination in terms of substitution functions $a_0 \sim a_1 \to P\,a_0 \to P\,a_1$ and contractibility of singleton types $(a_1 : A) \times (a_0 \sim a_1)$; see [7, Figure 2]. The path-connection algebra structure gives the latter, since using $\mathtt{ax_3}$ and $\mathtt{ax_4}$ we have

$$\begin{aligned} &\mathtt{ctr} : \{A : \mathcal{U}\}\{a_0\ a_1 : A\}(p : a_0 \sim a_1) \to (a_0, \mathtt{k}\,a_0) \sim (a_1, p) \\ &\mathtt{ctr}\,p\,i \triangleq (p\,i, \lambda j \to p(i \sqcap j)) \end{aligned} \tag{4}$$

However, to get suitably behaved substitution functions we have to consider families of types endowed with some extra structure; and that structure has to lift through the type-forming operations (products, functions, identity types, etc). This is what the definitions in the next section achieve.

4 Cohen-Coquand-Huber-Mörtberg (CCHM) Fibrations

In this section we show how to generalise the notion of fibration introduced in [11, Definition 13] from the particular presheaf model considered there to any topos with an interval object as in the previous section. To do so we introduce the notion of *cofibrant proposition* and use it to internalise the composition and filling operations described in [11].

4.1 Cofibrant propositions

Kan filling and other cofibrancy conditions on collections of subspaces have to do with lifting maps from a subspace to the whole space. Here we consider subspaces of spaces as subobjects of objects in toposes. Since subobjects are classified by morphisms to Ω, it is possible to consider collections of subobjects that are specified generically by giving a property of

[2] Here and elsewhere we use the Agda convention that braces $\{\}$ indicate implicit arguments.

propositions, in other words by giving a subobject $\mathtt{Cof} \rightarrowtail \Omega$. A monomorphism $m : A \rightarrowtail B$ is in the corresponding collection if its classifier $\lambda(y : B) \to \exists(x : A).\ m\,x = y : B \to \Omega$ factors through $\mathtt{Cof} \rightarrowtail \Omega$. We reduce lifting properties of morphisms out of the domains of such monomorphisms to extension properties of *partial elements* whose domains of definition are in $\mathtt{Cof}$. Recall that in intuitionistic logic, partial elelements of a type A are often represented by sub-singletons, that is, by terms $s : A \to \Omega$ satisfying $\forall(x\ x' : A).\ s\,x \wedge s\,x' \Rightarrow x = x'$. However, it will be more convenient to work with an extensionally equivalent representation as dependent pairs $\varphi : \Omega$ and $f : [\varphi] \to A$. The proposition φ is the *extent* of the partial element; in terms of sub-singletons it is equal to $\exists(x : A).\ s\,x$.

▶ **Definition 4.1** (Cofibrant partial elements, $\square\, A$)**.** We assume we are given a subobject $\mathtt{Cof} \rightarrowtail \Omega$ satisfying axioms $\mathtt{ax_5}$–$\mathtt{ax_8}$ in Figure 1. We call terms of type $\mathtt{Cof}$ *cofibrant propositions*. Given a type $A : \mathcal{U}$, we define the type of *cofibrant partial elements* of A to be

$$\square\, A \triangleq (\varphi : \mathtt{Cof}) \times ([\varphi] \to A) \tag{5}$$

An *extension* of such a partial element $(\varphi, f) : \square\, A$ is an element $a : A$ together with a proof of the following relation:

$$(\varphi, f) \nearrow a \triangleq \forall(u : [\varphi]).\ f\,u = a \tag{6}$$

We postpone discussing axiom $\mathtt{ax_8}$ until Section 5. Axioms $\mathtt{ax_5}$–$\mathtt{ax_7}$ give the simple properties of cofibrant propositions we use to define an internal notion of fibration generalising Definition 13 of [11] and show that it is closed under forming Σ-, Π- and $\mathtt{Id}$-types, as well as basic datatypes. In the figure $\mathtt{cof} : \Omega \to \Omega$ is the classifying morphism of the subobject $\mathtt{Cof} \rightarrowtail \Omega$, so that $\mathtt{Cof} = \{\varphi : \Omega \mid \mathtt{cof}\,\varphi\}$. The last of these three axioms, $\mathtt{ax_7}$, is equivalent to requiring that the collection of cofibrant monomorphisms is closed under composition (proof omitted); we only use this property in order to construct definitional identity types from propositional identity types (see Section 4.3).

Note that axioms $\mathtt{ax_2}$ and $\mathtt{ax_5}$ together imply that $\mathtt{cof}\ \bot$ holds, so that $\emptyset \rightarrowtail A$ is always a cofibrant monomorphism, where $\emptyset$ is the initial object. Axiom $\mathtt{ax_6}$ says that the union of two cofibrant subobjects is again cofibrant. This allows us to take the union of compatible cofibrant partial elements, which is used in many of the constructions below.

4.2 Composition and filling structures

Given an interval-indexed family of types $A : \mathtt{I} \to \mathcal{U}$, we think of elements of the dependent function type $\Pi_{\mathtt{I}} A \triangleq (i : \mathtt{I}) \to A\,i$ as dependently typed paths. We call elements of type $\square(\Pi_{\mathtt{I}} A)$ *cofibrant-partial paths*. Given $(\varphi, f) : \square(\Pi_{\mathtt{I}} A)$, we can evaluate it at a point $i : \mathtt{I}$ of the interval to get a cofibrant partial element $(\varphi, f)\ @\ i : \square(A\,i)$:

$$(\varphi, f)\ @\ i \triangleq (\varphi, \lambda(u : [\varphi]) \to f\,u\,i) \tag{7}$$

An operation for *filling from* $\mathtt{0}$ in $A : \mathtt{I} \to \mathcal{U}$ takes any $(\varphi, f) : \square(\Pi_{\mathtt{I}} A)$ and any $a_0 : A\,\mathtt{0}$ with $(\varphi, f)\ @\ \mathtt{0} \nearrow a_0$ and extends (φ, f) to a dependently typed path $g : \Pi_{\mathtt{I}} A$ with $g\,\mathtt{0} = a_0$. This is a form of uniform *Homotopy Extension and Lifting Property* (HELP) [28, Chapter 10, Section 3] stated internally in terms of cofibrant propositions rather than externally in terms of cofibrant monomorphisms, obviating the need to mention uniformity explicitly. Indeed a feature of the present work compared with Cohen *et al* is that the *uniformity* condition on composition/filling operations [11, Definition 13], which allows one to avoid the non-constructive aspects of the classical notion of Kan filling [6], becomes automatic when

the operations are formulated in terms of the internal collection $\mathtt{Cof}$ of cofibrant propositions. Since we are not assuming any structure on the interval for reversing paths (see Remark 3.1), we also need to consider the symmetric notion of filling from $\mathtt{1}$. So we have the following definition.

▶ **Definition 4.2** (Filling structures)**.** Let $\mathtt{\{0,1\}} \triangleq \{i : \mathtt{I} \mid i = \mathtt{0} \vee i = \mathtt{1}\}$; because of axiom $\mathtt{ax_2}$ this is isomorphic to the object of Booleans $(1 + 1)$ and hence there is a function $\overline{} : \mathtt{\{0,1\}} \to \mathtt{\{0,1\}}$ satisfying $\overline{\mathtt{0}} = \mathtt{1}$ and $\overline{\mathtt{1}} = \mathtt{0}$. Then the type of *filling structures* for $\mathtt{I}$-indexed families of types, $\mathtt{Fill} : (e : \mathtt{\{0,1\}})(A : \mathtt{I} \to \mathcal{U}) \to \mathcal{U}$, is defined by:

$$\begin{aligned}\mathtt{Fill}\, e\, A \triangleq (\varphi : \mathtt{Cof})(f : [\varphi] \to \Pi_{\mathtt{I}} A)(a : \{a' : A\, e \mid (\varphi, f) \mathbin{@} e \nearrow a'\}) \to \\ \{g : \Pi_{\mathtt{I}} A \mid (\varphi, f) \nearrow g \ \wedge\ g\, e = a\}\end{aligned} \tag{8}$$

A notable feature of [11] compared with preceding work [7] is that such filling structure can be constructed from a simpler *composition* structure that just produces an extension at one end of a cofibrant-partial path from an extension at the other end. We will deduce this using axioms $\mathtt{ax_3}$–$\mathtt{ax_6}$ after defining the main notion of this paper.

▶ **Definition 4.3** (The CwF of CCHM fibrations)**.** A *CCHM fibration* (A, α) over a type $\Gamma : \mathcal{U}$ is a family $A : \Gamma \to \mathcal{U}$ equipped with a fibration structure $\alpha : \mathtt{Fib}\, A$, where $\mathtt{Fib} : \{\Gamma : \mathcal{U}\}(A : \Gamma \to \mathcal{U}) \to \mathcal{U}$ is defined by

$$\mathtt{Fib}\, \{\Gamma\}\, A \triangleq (e : \mathtt{\{0,1\}})(p : \mathtt{I} \to \Gamma) \to \mathtt{Comp}\, e\, (A \circ p) \tag{9}$$

Here $\mathtt{Comp} : (e : \mathtt{\{0,1\}})(A : \mathtt{I} \to \mathcal{U}) \to \mathcal{U}$ is the type of *composition structures* for $\mathtt{I}$-indexed families:

$$\begin{aligned}\mathtt{Comp}\, e\, A \triangleq (\varphi : \mathtt{Cof})(f : [\varphi] \to \Pi_{\mathtt{I}} A) \to \\ \{a_0 : A\, e \mid (\varphi, f) \mathbin{@} e \nearrow a_0\} \to \{a_1 : A\, \overline{e} \mid (\varphi, f) \mathbin{@} \overline{e} \nearrow a_1\}\end{aligned} \tag{10}$$

Unwinding the definition, if $\alpha : \mathtt{Fib}\, A$ then $\alpha\, \mathtt{0}$ satisfies that for each cofibrant partial path $f : [\varphi] \to \Pi_{\mathtt{I}}(A \circ p)$ over a path $p : \mathtt{I} \to \Gamma$, if $a_0 : A\, \mathtt{0}$ extends the partial element $(\varphi, f) \mathbin{@} \mathtt{0}$, (that is, $\forall (u : [\varphi]).\ f\, u\, \mathtt{0} = a_0$), then $\alpha\, \mathtt{0}\, p\, \varphi\, f\, a_0 : A\, \mathtt{1}$ extends $(\varphi, f) \mathbin{@} \mathtt{1}$, that is $\forall (u : [\varphi]).\ f\, u\, \mathtt{1} = \alpha\, \mathtt{0}\, p\, \varphi\, f\, a_0$; and similarly for $\alpha\, \mathtt{1}$.

CCHM fibrations are closed under re-indexing: given $\gamma : \Delta \to \Gamma$ and $A : \Gamma \to \mathcal{U}$, we get a function $[_]\gamma : \mathtt{Fib}\, A \to \mathtt{Fib}(A \circ \gamma)$ defined by $[\alpha]\gamma\, e\, p \triangleq \alpha\, e\, (\gamma \circ p)$. It follows that we get the structure of a Category with Families by taking families to be CCHM fibrations (A, α) over each $\Gamma : \mathcal{U}$ and elements of such a family to be dependent functions in $(x : \Gamma) \to A\, x$.

▶ Remark 4.4 (Fibrant objects). We say $A : \mathcal{U}$ is a *fibrant object* if we have a fibration structure for the constant family $\lambda(_ : 1) \to A$ over the terminal object 1. Note that if $A : \Gamma \to \mathcal{U}$ has a fibration structure, then for each $x : \Gamma$ the type $A\, x : \mathcal{U}$ is fibrant. However the converse is not true: having a family of fibration structures, that is, an element of $(x : \Gamma) \to \mathtt{Fib}(\lambda(_ : 1) \to A\, x)$, is weaker than having a fibration structure for $A : \Gamma \to \mathcal{U}$. For example, having a fibration structure allows one to transport elements along paths in Γ (see the $\mathtt{subst}$ functions defined below in (19)), whereas clearly a family of fibrant objects may not possess such transport operations.

If $\alpha : \mathtt{Fill}\, e\, A$, then $\lambda \varphi\, f\, a \to \alpha\, \varphi\, f\, a\, \overline{e} : \mathtt{Comp}\, e\, A$ and so every filling structure gives rise to a composition structure. Conversely, the composition structure of a CCHM fibration gives rise to filling structure:

▶ **Lemma 4.5.** *Given* $\Gamma : \mathcal{U}$, $A : \Gamma \to \mathcal{U}$, $e : \{\mathtt{0},\mathtt{1}\}$, $\alpha : \mathtt{Fib}\, A$ *and* $p : \mathtt{I} \to \Gamma$, *there is a filling structure* $\mathtt{fill}\, e\, \alpha\, p : \mathtt{Fill}\, e\, (A \circ p)$ *that agrees with* α *at* $\overline{e}$, *that is:*

$$\forall(\varphi : \mathtt{Cof})(f : [\varphi] \to \Pi_{\mathtt{I}} A)(a : A(p\, e)).\ (\varphi, f)\, @\, e \nearrow a \ \Rightarrow\ \mathtt{fill}\, e\, \alpha\, p\, \varphi\, f\, a\, \overline{e} = \alpha\, e\, p\, \varphi\, f\, a \quad (11)$$

Proof. First note that if two partial elements $f : [\varphi] \to A$ and $g : [\psi] \to A$ are *compatible*, that is, if the following relation holds

$$(\varphi, f) \smile (\psi, g) \triangleq \forall(u : [\varphi])(v : [\psi]).\ f\, u = g\, v \quad (12)$$

then by axiom $\mathtt{ax_6}$ their union $f \cup g : [\varphi \vee \psi] \to A$ gives a cofibrant partial element provided that (φ, f) and (ψ, g) are cofibrant partial elements. We use this in a construction of filling from composition that follows [11, Section 4.4], but just using the path-connection algebra structure on $\mathtt{I}$, rather than a de Morgan algebra structure. Suppose $\Gamma : \mathcal{U}$, $A : \Gamma \to \mathcal{U}$, $e : \{\mathtt{0},\mathtt{1}\}$, $\alpha : \mathtt{Fib}\, A$, $p : \mathtt{I} \to \Gamma$, $\varphi : \mathtt{Cof}$, $f : [\varphi] \to \Pi_{\mathtt{I}}(A \circ p)$, $a : A(p\, e)$ with $(\varphi, f)\, @\, e \nearrow a$, and $i : \mathtt{I}$. Then we can define

$$\begin{aligned}
&\mathtt{fill}\, e\, \alpha\, p\, \varphi\, f\, a\, i \triangleq \alpha\, e\, (p'\, i)\, (\varphi \vee i = e)\, (f'\, i \cup g\, i)\, a, \text{ where} \quad (13)\\
&p' : \mathtt{I} \to \mathtt{I} \to \Gamma \text{ is defined by } p'\, i\, j \triangleq p(i \sqcap_e j)\\
&f' : (i : \mathtt{I}) \to [\varphi] \to \Pi_{\mathtt{I}}(A \circ (p'\, i)) \text{ is defined by } f'\, i\, u\, j \triangleq f\, u\, (i \sqcap_e j)\\
&g : (i : \mathtt{I}) \to \{g' : [i = e] \to \Pi_{\mathtt{I}}(A \circ (p' i)) \mid (\varphi, f'\, i) \smile (i = e, g')\} \text{ is defined by } g\, i\, v\, j \triangleq a
\end{aligned}$$

and where $\sqcap_e$ is given by $\sqcap_0 \triangleq \sqcap$ and $\sqcap_1 \triangleq \sqcup$. We omit the proof that the above definition of $\mathtt{fill}$ has the required properties. ◀

Compared with [7], the fact that filling can be defined from composition considerably simplifies the process of lifting fibration structure through the usual type-forming constructs; for example:

▶ **Theorem 4.6** (Fibrant Σ- and Π-types)**.** *There are functions*

$$\begin{aligned}
&\mathtt{Fib}_\Sigma : \{\Gamma : \mathcal{U}\}\{A_1 : \Gamma \to \mathcal{U}\}\{A_2 : (x : \Gamma) \times A_1\, x \to \mathcal{U}\} \to \quad (14)\\
&\qquad \mathtt{Fib}\, A_1 \to \mathtt{Fib}\, A_2 \to \mathtt{Fib}(\Sigma\, A_1\, A_2)\\
&\mathtt{Fib}_\Pi : \{\Gamma : \mathcal{U}\}\{A_1 : \Gamma \to \mathcal{U}\}\{A_2 : (x : \Gamma) \times A_1\, x \to \mathcal{U}\} \to \quad (15)\\
&\qquad \mathtt{Fib}\, A_1 \to \mathtt{Fib}\, A_2 \to \mathtt{Fib}(\Pi\, A_1\, A_2)
\end{aligned}$$

where $\Sigma\, A_1\, A_2\, x \triangleq (a_1 : A_1\, x) \times A_2(x, a_1)$ *and* $\Pi\, A_1\, A_2\, x \triangleq (a_1 : A_1\, x) \to A_2(x, a_1)$. *These functions are stable under re-indexing. Hence the category with families given by CCHM fibrations has* Σ- *and* Π-*types.*

Proof. The proof uses the above lemma and constructions similar to those in [11, Section 4.5]. We just give the construction of $\mathtt{Fib}_\Pi$ here, to show how to avoid the use Cohen *et al* make of de Morgan involution. Given $\Gamma : \mathcal{U}$, $A_1 : \Gamma \to \mathcal{U}$, $A_2 : (x : \Gamma) \times A_1\, x \to \mathcal{U}$, $\alpha_1 : \mathtt{Fib}\, A_1$, $\alpha_2 : \mathtt{Fib}\, A_2$, $e : \{\mathtt{0},\mathtt{1}\}$, $p : \mathtt{I} \to \Gamma$, $\varphi : \mathtt{Cof}$, $f : [\varphi] \to \Pi_{\mathtt{I}}((\Pi\, A_1\, A_2) \circ p)$, $g : (\Pi\, A_1\, A_2)(p\, e)$ with

$(\varphi, f) \mathbin{@} e \nearrow g$ and $a_1 : A_1(p\,\overline{e})$, using Lemma 4.5 we define

$$\begin{aligned}
&\mathtt{Fib}_\Pi\ \alpha_1\,\alpha_2\,e\,p\,\varphi\,f\,g\,a_1 \triangleq \alpha_2\,e\,q\,\varphi\,f_2\,a_2, \text{ where} \\
&\quad f_1 : \Pi_{\mathtt{I}}(A_1 \circ p) \\
&\quad f_1 \triangleq \mathtt{fill}\,\overline{e}\,\alpha_1\,p\,\bot\,\mathtt{elim}_\emptyset\ a_1 \\
&\quad q : \mathtt{I} \to (x : \Gamma) \times A_1\,x \\
&\quad q \triangleq \langle p\ ,\ f_1 \rangle \\
&\quad f_2 : [\varphi] \to \Pi_{\mathtt{I}}(A_2 \circ q) \\
&\quad f_2\,u\,i \triangleq f\,u\,i\,(f_1\,i) \\
&\quad a_2 : \{a_2' : A_2(q\,e) \mid (\varphi, f_2) \mathbin{@} e \nearrow a_2'\} \\
&\quad a_2 \triangleq g(f_1\,e)
\end{aligned} \tag{16}$$

where for any $B : \mathcal{U}$, $\mathtt{elim}_\emptyset : [\bot] \to B$ denotes the unique function given by initiality of $[\bot]$, so that $(\bot, \mathtt{elim}_\emptyset) : \Box B$ because $\bot : \mathtt{Cof}$ by $\mathtt{ax_2}$ and $\mathtt{ax_5}$; and furthermore $(\bot, \mathtt{elim}_\emptyset) \nearrow b$ holds for any $b : B$. ◀

The theorem allows us to construct fibration structures for Σ- and Π-types, given fibration structures for their constituent types. But are there any fibration structures to begin with? We answer this question by showing that the natural number object $\mathtt{N}$ in the topos is always fibrant. This is proved for the topos of cubical sets $\hat{\mathcal{C}}$ in [7, Section 4.5] by defining a composition structure by primitive recursion. We give a more elementary proof using the fact that the interval object in $\hat{\mathcal{C}}$ satisfies axiom $\mathtt{ax_1}$ (see Theorem 6.1).

▶ **Theorem 4.7** ($\mathtt{N}$ is fibrant). *If $\mathtt{N}$ is an object with decidable equality, then there is a function $\mathtt{Fib}_{\mathtt{N}} : \{\Gamma : \mathcal{U}\} \to \mathtt{Fib}(\lambda(_ : \Gamma) \to \mathtt{N})$. In particular, if the topos $\mathcal{E}$ has a natural number object $1 \xrightarrow{\mathtt{z}} \mathtt{N} \xrightarrow{\mathtt{s}} \mathtt{N}$, then the category with families given by CCHM fibrations has a natural number object.*

Proof. Suppose $\Gamma : \mathcal{U}$, $e : \mathtt{\{0,1\}}$, $p : \mathtt{I} \to \Gamma$, $\varphi : \mathtt{Cof}$, $f : [\varphi] \to \Pi_{\mathtt{I}}(\lambda_ \to \mathtt{N})$ and $n : \mathtt{N}$ with $(\varphi, f) \mathbin{@} e \nearrow n$. By assumption on $\mathtt{N}$, for each $u : [\varphi]$ the property $\lambda(i : \mathtt{I}) \to (f\,u\,i = n) : \mathtt{I} \to \Omega$ is decidable; hence by axiom $\mathtt{ax_1}$ and the fact that $f\,u\,e = n$, we also have $f\,u\,\overline{e} = n$. Therefore we can get $\mathtt{Fib}_{\mathtt{N}}\,e\,p\,\varphi\,f\,n : \{n' : \mathtt{N} \mid (\varphi, f) \mathbin{@} \overline{e} \nearrow n'\}$ just by defining: $\mathtt{Fib}_{\mathtt{N}}\,e\,p\,\varphi\,f\,n \triangleq n$. For the last part of the theorem we use the fact that in a topos with natural number object, equality of numbers is decidable. ◀

A similar use of axiom $\mathtt{ax_1}$ suffices to prove:

▶ **Theorem 4.8** (Fibrant coproducts). *Writing $A_1 \xrightarrow{\mathtt{inl}} A_1 + A_2 \xleftarrow{\mathtt{inr}} A_2$ for the coproduct of A_1 and A_2 in $\mathcal{E}$, we lift this to families of types, $_ \uplus _ : \{\Gamma : \mathcal{U}\}(A_1\ A_2 : \Gamma \to \mathcal{U}) \to \Gamma \to \mathcal{U}$, by defining $(A_1 \uplus A_2)\,x \triangleq A_1\,x + A_2\,x$. Then there is a function*

$$\mathtt{Fib}_\uplus : \{\Gamma : \mathcal{U}\}\{A_1\ A_2 : \Gamma \to \mathcal{U}\} \to \mathtt{Fib}\,A_1 \to \mathtt{Fib}\,A_2 \to \mathtt{Fib}(A_1 \uplus A_2) \tag{17}$$

and this fibration structure on coproducts is stable under re-indexing. Hence the category with families given by CCHM fibrations has coproducts. ◀

4.3 Identity types

The next result follows from axioms $\mathtt{ax_2}$–$\mathtt{ax_6}$ by a construction like that in [7, Section 4.5].

▶ **Theorem 4.9** (Fibrant path types). *There is a function* $\mathtt{Fib}_{\mathtt{Path}} : \{\Gamma : \mathcal{U}\}\{A : \Gamma \to \mathcal{U}\} \to \mathtt{Fib}\, A \to \mathtt{Fib}(\mathtt{Path}\, A)$*, where* $\mathtt{Path}\, A : (x : \Gamma) \times (A\, x \times A\, x) \to \mathcal{U}$ *is given by*

$$\mathtt{Path}\, A\, (x, (a_0, a_1)) \triangleq a_0 \sim a_1 \tag{18}$$

and where $\sim$ *is as in* (2)*. This fibration structure on path types is stable under re-indexing.* ◀

These path types in the CwF of CCHM fibrations (Definition 4.3) satisfy the Coquand formulation of identity types with propositional computation properties [7, Figure 2]. Thus in addition to the contractibility of singleton types (4), we get *substitution functions* for transporting elements of a fibration along a path

$$\begin{aligned} &\mathtt{subst} : \{\Gamma : \mathcal{U}\}\{A : \Gamma \to \mathcal{U}\}\{\alpha : \mathtt{Fib}\, A\}\{x_0\ x_1 : \Gamma\} \to (x_0 \sim x_1) \to A\, x_0 \to A\, x_1 \\ &\mathtt{subst}\, p\, a \triangleq \alpha\, \mathsf{0}\, p\, \bot\, \mathtt{elim}_{\emptyset}\ a \end{aligned} \tag{19}$$

using the cofibrant partial elements $(\bot, \mathtt{elim}_{\emptyset})$ mentioned in the proof of Theorem 4.6. By Lemma 4.5 we have that these substitution functions satisfy a propositional computation rule for constant paths (3):

$$\begin{aligned} &\mathtt{H} : \{\Gamma : \mathcal{U}\}\{A : \Gamma \to \mathcal{U}\}\{\alpha : \mathtt{Fib}\, A\}\{x : \Gamma\}(a : A\, x) \to (a \sim \mathtt{subst}(\mathtt{k}\, x)\, a) \\ &\mathtt{H}\, a \triangleq \mathtt{fill}\, \mathsf{0}\, \alpha\, (\mathtt{k}\, x)\, \bot\, \mathtt{elim}_{\emptyset}\ a \end{aligned} \tag{20}$$

To get Martin-Löf identity types with standard definitional, rather than propositional computation properties from these path types, we can use a version of Swan's construction [32] like the one in Section 9.1 of [11], but only using the path-connection algebra structure on $\mathtt{I}$, rather than a de Morgan algebra structure. This is the only place that axiom $\mathtt{ax}_7$ is used; we need the fact that the universe given by $\mathtt{Cof}$ and $[_] : \mathtt{Cof} \to \mathcal{U}$ is closed under dependent products:

▶ **Lemma 4.10.** *The following term of type* Ω *is provable:* $\forall(\varphi : \Omega)(f : [\varphi] \to \Omega).\, \mathtt{cof}\, \varphi \Rightarrow (\forall(u : [\varphi]).\ \mathtt{cof}(f\, u)) \Rightarrow \mathtt{cof}(\exists(u : [\varphi]).\, f\, u)$*.*

Proof. Note that if $u : [\varphi]$ then $(\exists(v : [\varphi]).\, f\, v) = f\, u$ and hence $\mathtt{cof}(\exists(v : [\varphi]).\, f\, v) = \mathtt{cof}(f\, u)$. So $\forall(u : [\varphi]).\ \mathtt{cof}(f\, u)$ equals $\varphi \Rightarrow \mathtt{cof}(\exists(v : [\varphi]).\, f\, v)$. Therefore from $\mathtt{cof}\, \varphi$ and $\forall(u : [\varphi]).\ \mathtt{cof}(f\, u)$ by axiom $\mathtt{ax}_7$ we get $\mathtt{cof}(\varphi \wedge \exists(v : [\varphi]).\, f\, v)$ and hence $\mathtt{cof}(\exists(v : [\varphi]).\, f\, v)$, since $(\exists(v : [\varphi]).\, f\, v) \Rightarrow \varphi$. ◀

▶ **Theorem 4.11** (Fibrant identity types). *Define identity types by:*

$$\begin{aligned} &\mathtt{Id} : \{\Gamma : \mathcal{U}\}(A : \Gamma \to \mathcal{U}) \to (x : \Gamma) \times (A\, x \times A\, x) \to \mathcal{U} \\ &\mathtt{Id}\, A\, (x, (a_0, a_1)) \triangleq (p : \mathtt{Path}\, A\, (x, (a_0, a_1))) \times \{\varphi : \mathtt{Cof} \mid \varphi \Rightarrow \forall(i : \mathtt{I}).\, p\, i = a_0\} \end{aligned} \tag{21}$$

Then there is a function $\mathtt{Fib}_{\mathtt{Id}} : \{\Gamma : \mathcal{U}\}\{A : \Gamma \to \mathcal{U}\} \to \mathtt{Fib}\, A \to \mathtt{Fib}(\mathtt{Id}\, A)$ *and the fibrations* $(\mathtt{Id}\, A, \mathtt{Fib}_{\mathtt{Id}}\, A)$ *can be given the structure of Martin-Löf identity types in the CwF of CCHM fibrations.*

Proof. Given $\Gamma : \mathcal{U}$, $A : \Gamma \to \mathcal{U}$ and $\alpha : \mathtt{Fib}\, A$, using Theorems 4.6 and 4.9 we define $\mathtt{Fib}_{\mathtt{Id}}\, \alpha \triangleq \mathtt{Fib}_{\Sigma}(\mathtt{Fib}_{\mathtt{Path}}\, \alpha)\, \beta$, where $\beta : \mathtt{Fib}\, \Phi$ with

$$\begin{aligned} &\Phi : (y : (x : \Gamma) \times (A\, x \times A\, x)) \times \mathtt{Path}\, A\, y \to \mathcal{U} \\ &\Phi((x, (a_0, a_1)), p) \triangleq \{\varphi : \mathtt{Cof} \mid \varphi \Rightarrow \forall(i : \mathtt{I}).\, p\, i = a_0\} \end{aligned}$$

and the fibration structure β mapping $e : \mathtt{\{0,1\}}$, $p : \mathtt{I} \to (y : (x : \Gamma) \times (A\,x \times A\,x)) \times \mathtt{Path}\,A\,y$, $\varphi : \mathtt{Cof}$, $f : [\varphi] \to \Pi_{\mathtt{I}}(\Phi \circ p)$ and $\varphi' : \Phi(p\,e)$ with $(\varphi, f)\,@\,e \nearrow \varphi'$ to the term $\beta\,e\,p\,\varphi\,f\,\varphi' \triangleq \exists(u : [\varphi]).\ f\,u\,\overline{e}$ (using Lemma 4.10 to see that this is well defined). We get the usual introduction, elimination and computation rules for these identity types as follows. Since $\top : \mathtt{Cof}$ holds by axiom $\mathtt{ax_5}$, identity introduction $\mathtt{refl} : \{\Gamma : \mathcal{U}\}\{A : \Gamma \to \mathcal{U}\}\{x : \Gamma\}(a : A\,x) \to \mathtt{Id}\,A\,(x, (a, a))$ can be defined by $\mathtt{refl}\,a \triangleq (\lambda a\,i \to a, \top)$. Identity elimination

$$\begin{aligned}\mathtt{J} : \{\Gamma : \mathcal{U}\}(A : \Gamma \to \mathcal{U})(x : \Gamma)(a_0 : A\,x)(B : (a : A\,x) \times \mathtt{Id}\,A\,(x, (a_0, a)) \to \mathcal{U}) \\ (\beta : \mathtt{Fib}\,B)(a_1 : A\,x)(e : \mathtt{Id}\,A\,(x, (a_0, a_1))) \to B(a_0, \mathtt{refl}\,a_0) \to B(a_1, e)\end{aligned} \tag{22}$$

is given by $\mathtt{J}\,A\,x\,a_0\,B\,\beta\,a_1(p, \varphi)\,b \triangleq \beta\,\mathtt{0}\,\langle p\ ,\ q\rangle\,\varphi\,f\,b$ where $q : (i : \mathtt{I}) \to \mathtt{Id}\,A\,x\,(a_0, p\,i)$ is $q\,i\,j \triangleq (p(i \sqcap_{\mathtt{0}} j), \varphi \vee i = \mathtt{0})$ and $f : [\varphi] \to \Pi_{\mathtt{I}}(B \circ \langle p\ ,\ q\rangle)$ is $f\,u\,i \triangleq b$. ◀

5 Towards Univalence

Voevodsky's *univalence axiom* [33, Section 2.10] for a universe $\mathcal{V}$ in a CwF (with Σ-, Π- and $\mathtt{Id}$-types) states that for every $A, B : \mathcal{V}$ the canonical function from $\mathtt{Id}\,\mathcal{V}\,A\,B$ to $(f : A \to B) \times \mathtt{Equiv}\,f$ is an equivalence. The notion of *equivalence* can be defined in terms of having contractible homotopy fibres [33, Section 4.4]:

$$\mathtt{Contr} : \mathcal{U} \to \mathcal{U} \tag{23}$$

$$\mathtt{Contr}\,A \triangleq (a_0 : A) \times ((a : A) \to a_0 \sim a)$$

$$\mathtt{Equiv} : \{A\ B : \mathcal{U}\}(f : A \to B) \to \mathcal{U} \tag{24}$$

$$\mathtt{Equiv}\,f \triangleq (b : B) \to \mathtt{Contr}((a : A) \times f\,a \sim b)$$

Cohen *et al* construct such a universe in the (CwF associated to the) presheaf topos of cubical sets by adapting the Hofmann-Streicher universe construction for presheaf categories [20]. We currently have no method for expressing this in the internal type theory of a general topos. Nevertheless in this section we present constructions using a *glueing* construction as in [11] that we conjecture suffice to ensure that if a universe of CCHM fibrations exists, then it satisfies most, if not all, of the conditions of the univalence axiom. Specifically we show how to transform equivalences into paths and vice-versa just for fibrant objects, rather than for fibrant families of objects (cf. Remark 4.4). Were there to be a universe of fibrations, then a proof of equality between types A and B in that universe would be a family $P : \mathtt{I} \to \mathcal{U}$ that not only satisfies $P\,\mathtt{0} = A$ and $P\,\mathtt{1} = B$, but also is fibrant. Note that if P has a fibration structure, then A and B are necessarily fibrant objects. We show that given such a family P it is always possible to construct an equivalence $f : A \to B$. Conversely, given an equivalence $f : A \to B$ between fibrant objects, it is always possible to construct such a P, provided cofibrant propositions satisfy a certain strictness property (axiom $\mathtt{ax_8}$ in Figure 1).

We begin by defining a path type for elements of the universe $\mathcal{U}$, in the style of (2). To do this we assume a second universe $\mathcal{U}_1$ with $\mathcal{U} : \mathcal{U}_1$. We sometimes refer to terms of type $\mathcal{U}$ as *small* types and terms of type $\mathcal{U}_1$ as *large* types. Define $_ \sim_{\mathcal{U}} _ : \mathcal{U} \to \mathcal{U} \to \mathcal{U}_1$ by

$$A \sim_{\mathcal{U}} B \triangleq \{P : \mathtt{I} \to \mathcal{U} \mid P\,\mathtt{0} = A \wedge P\,\mathtt{1} = B\} \tag{25}$$

▶ **Theorem 5.1.** *There is a function*

$$\mathtt{pathToEquiv} : \{A\ B : \mathcal{U}\}(P : A \sim_{\mathcal{U}} B)(\rho : \mathtt{Fib}\,P) \to (f : A \to B) \times \mathtt{Equiv}\,f \tag{26}$$

Proof. Define maps $f : A \to B$ and $g : B \to A$ as follows:

$$f\,a \triangleq \rho\,\mathtt{0}\;id \perp \mathtt{elim}_{\emptyset}\;a \qquad\qquad g\,b \triangleq \rho\,\mathtt{1}\;id \perp \mathtt{elim}_{\emptyset}\;b$$

This definition is well-typed since $P\,\mathtt{0} = A$ and $P\,\mathtt{1} = B$. Since both functions are defined using composition structure, for every $a : A$ and $b : B$ we can use filling (Lemma 4.5) to find dependently typed paths

$$p, q : \Pi_{\mathtt{I}} P \text{ with } p\,\mathtt{0} = a,\ p\,\mathtt{1} = f\,a,\ q\,\mathtt{0} = b \text{ and } q\,\mathtt{1} = g\,b \tag{27}$$

Since $P : \mathtt{I} \to \mathcal{U}$ has a fibration structure, A and B are fibrant objects. Therefore as in Section 4.3, we have path types for them satisfying the properties of identity types propositionally. So it is possible to combine the paths (27) to get $a \sim g\,(f\,a)$ and $b \sim f\,(g\,b)$. Hence f and g are quasi-inverses [33, Section 4.1] and hence in particular f is an equivalence. ◀

We now wish to construct a map going the other way, from equivalences to paths in the universe. To do so we use the notion of cofibrant-partial families of types: given a type $\Gamma : \mathcal{U}$ and a cofibrant property $\Phi : \Gamma \to \mathtt{Cof}$, we define $(\Gamma, \Phi) : \mathcal{U}$ by

$$\Gamma\,,\,\Phi \triangleq (x : \Gamma) \times [\Phi\,x] \tag{28}$$

and say that a term of type $A : (\Gamma\,,\,\Phi) \to \mathcal{U}$ is a *cofibrant-partial family of types* over Γ. Next we give a version of the *glueing* construction [11, Section 6], which allows one to extend such a cofibrant-partial family along a function $f : (x : \Gamma)(v : [\Phi\,x]) \to A(x, v) \to B\,x$ to a total family over Γ.

▶ **Definition 5.2** (Glueing). Given $\Gamma : \mathcal{U}$, $\Phi : \Gamma \to \mathtt{Cof}$, $A : \Gamma\,,\,\Phi \to \mathcal{U}$, $B : \Gamma \to \mathcal{U}$ and $f : (x : \Gamma)(v : [\Phi\,x]) \to A(x, v) \to B\,x)$, define:

$$\mathtt{Glue}\,\Phi\,A\,B\,f : \Gamma \to \mathcal{U} \tag{29}$$

$$\mathtt{Glue}\,\Phi\,A\,B\,f\,x \triangleq (g : (v : [\Phi\,x]) \to A(x, v)) \times \{b : B\,x \mid \forall (v : [\Phi\,x]).\ f\,x\,v\,(g\,v) = b\}$$

$$\mathtt{glue}\,f : ((x, u) : \Gamma\,,\,\Phi) \to A(x, u) \to \mathtt{Glue}\,\Phi\,A\,B\,f\,x \tag{30}$$

$$\mathtt{glue}\,f\,(x, u)\,a \triangleq (\lambda(_ : [\Phi\,x]) \to a,\ f\,x\,u\,a)$$

$$\mathtt{unglue}\,f : (x : \Gamma) \to \mathtt{Glue}\,\Phi\,A\,B\,f\,x \to B\,x \tag{31}$$

$$\mathtt{unglue}\,f\,x \triangleq \mathtt{snd}$$

▶ **Theorem 5.3.** $\mathtt{Glue}\,\Phi\,A\,B\,f$ *has a fibration structure if* A *and* B *have one and if* f *has the structure of an equivalence. In other words there is a function*

$$\begin{aligned}\mathtt{Fib}_{\mathtt{Glue}} : {} & \{\Gamma : \mathcal{U}\}\{\Phi : \Gamma \to \mathtt{Cof}\}\{A : \Gamma\,,\,\Phi \to \mathcal{U}\}\{B : \Gamma \to \mathcal{U}\} \\ & (f : (x : \Gamma)(u : [\Phi\,x]) \to A(x, u) \to B\,x) \to \\ & ((x : \Gamma)(v : [\Phi\,x]) \to \mathtt{Equiv}(f\,x\,v)) \to \mathtt{Fib}\,A \to \mathtt{Fib}\,B \to \mathtt{Fib}(\mathtt{Glue}\,\Phi\,A\,B\,f)\end{aligned} \tag{32}$$

Proof. In outline, our proof of the theorem is as follows:

- Characterise equivalences in terms of a notion of *extension structure* (cf. Lemma 7 in [11]).
- Show that $\mathtt{unglue}\,f\,x : \mathtt{Glue}\,\Phi\,A\,B\,f\,x \to B\,x$ has such an extension structure when each $f\,x\,v$ does and when B is a CCHM fibration.
- Show that for a family of functions with an extension structure, if the codomain has a fibration structure, then so does the domain. Applying this to $\mathtt{unglue}\,f$, we get that $\mathtt{Glue}\,\Phi\,A\,B\,f$ has a fibration structure.

The details can be found in our Agda development. This proof differs from that in [11] in that it does not need cofibrant propositions to be closed under $\mathtt{I}$-indexed conjunction (cf. the $\forall$ quantifier defined in Section 4.1 of [11]). However, unlike in [11], the construction does not yield an element $\mathtt{Fib}_{\mathtt{Glue}}\, f\, \varepsilon\, \alpha\, \beta$ that restricts to α on Φ – a property that is probably needed for the construction of a univalent universe. ◀

We now have a way to interpret the glueing operation from [11] that meets some of the necessary requirements; see [11, Figure 4]. However, the current construction does not have certain *strictness* properties. In particular cubical type theory requires that, when restricted to a context where Φ holds, glueing should be equal to A "on the nose", that is that for any $x : \Gamma$ with $u : [\Phi\, x]$, we should have $\mathtt{Glue}\ \Phi\, A\, B\, f\, x = A(x, u)$. To satisfy such a requirement we postulate a further axiom $\mathtt{ax_8}$ that allows us to extend a partial type along an *isomorphism*

$$A \cong B \triangleq \{f : A \to B \mid (\exists g : B \to A)\, (g \circ f = id) \wedge (f \circ g = id)\} \tag{33}$$

to get a total type. Isomorphisms have inverses up to the extensional equality of the internal type theory, in contrast to equivalences which only have inverses up to path equality. Axiom $\mathtt{ax_8}$ says that any partial type A that is isomorphic to a total type B everywhere that A is defined, can be extended to a total type B' that is isomorphic to B.

▶ **Definition 5.4** (Strict Glueing)**.** Given $\Gamma : \mathcal{U}$, $\Phi : \Gamma \to \mathtt{Cof}$, $A : \Gamma\, , \Phi \to \mathcal{U}$, $B : \Gamma \to \mathcal{U}$ and $f : (x : \Gamma)(v : [\Phi\, x]) \to A(x, v) \to B\, x$, define $\mathtt{SGlue}\, \Phi\, A\, B\, f : \Gamma \to \mathcal{U}$ by

$$\mathtt{SGlue}\, \Phi\, A\, B\, f\, x \triangleq$$
$$\mathtt{fst}(\mathtt{ax_8}\, (\lambda u : [\Phi\, x] \to A(x, u))\, (\mathtt{Glue}\ \Phi\, A\, B\, f\, x)\, (\lambda u : [\Phi\, x] \to \mathtt{glue}\, f\, (x, u))) \tag{34}$$

Note that $\mathtt{SGlue}$ has the desired strictness property: given any $(x, u) : \Gamma\, , \Phi$, by $\mathtt{ax_8}$ we have $A(x, u) = \mathtt{fst}(\mathtt{ax_8}\, (\lambda u : [\Phi\, x] \to A(x, u))\, (\mathtt{Glue}\ \Phi\, A\, B\, f\, x)\, (\lambda u : [\Phi\, x] \to \mathtt{glue}\, f\, (x, u)))$ and hence

$$\forall (x : \Gamma)(u : [\Phi\, x]).\ \mathtt{SGlue}\, \Phi\, A\, B\, f\, x = A(x, u) \tag{35}$$

▶ **Theorem 5.5.** $\mathtt{SGlue}$ *has a fibration structure if A and B have one and f has the structure of an equivalence.*

Proof. It is easy to show that (fibrewise) isomorphisms preserve fibration structures. Hence we obtain a fibration structure on $\mathtt{SGlue}$ by transporting the structure obtained from $\mathtt{Fib}_{\mathtt{Glue}}$ (Theorem 5.3) along the isomorphism from $\mathtt{ax_8}$. ◀

We are now able to construct a map from equivalences to paths in the universe:

▶ **Theorem 5.6.** *There is a function*

$$\begin{aligned}\mathtt{equivToPath} : \{A\ B : \mathcal{U}\}\{\alpha : \mathtt{Fib}(\lambda_ : 1 \to A)\}\{\beta : \mathtt{Fib}(\lambda_ : 1 \to B)\} \\ (f : A \to B) \to (\mathtt{Equiv}\, f) \to (P : A \sim_{\mathcal{U}} B) \times (\mathtt{Fib}\, P)\end{aligned} \tag{36}$$

Proof. Define the following:

$\Phi : \mathtt{I} \to \mathtt{Cof}$

$\Phi\, i \triangleq (i = 0) \vee (i = 1)$

$C : (\mathtt{I}\, , \Phi) \to \mathcal{U}$

$C\, (i, u) \triangleq ((\lambda_ : [i = 0] \to A) \cup (\lambda_ : [i = 1] \to B))\, u$

$f' : (i : \mathtt{I})(u : [\Phi\, i]) \to C(i, u) \to B$

$f'\, i \triangleq (\lambda_ : [i = 0] \to f) \cup (\lambda_ : [i = 1] \to id)$

Now let $P \triangleq \mathtt{SGlue}\ \Phi\, C\, (\lambda_ : \mathtt{I} \to B)\, f'$ and observe that $P\,\mathtt{0} = A$ and $P\,\mathtt{1} = B$ by the strictness property of $\mathtt{SGlue}$. Further, we can show that f' is an equivalence since f and the identity are both equivalences; and using α, β and $\mathtt{ax}_1$, we can define a fibration structure on C. Hence, by Theorem 5.5, we get a fibration structure on P. ◀

The following theorem shows that for fibrant objects the functions $\mathtt{pathToEquiv}$ and $\mathtt{equivToPath}$ are mutually inverse up to path equality. We omit its proof here.

▶ **Theorem 5.7.** *Given $A, B : \mathcal{U}$, define*

$$\mathtt{pathToPath} : (P : A \sim_{\mathcal{U}} B)(\rho : \mathtt{Fib}\, P) \to (P' : A \sim_{\mathcal{U}} B) \times (\rho' : \mathtt{Fib}\, P') \tag{37}$$

$$\mathtt{pathToPath}\, P\, \rho \triangleq \mathtt{equivToPath}\, (\mathtt{fst}(\mathtt{pathToEquiv}\ P\, \rho))\, (\mathtt{snd}(\mathtt{pathToEquiv}\ P\, \rho))$$

$$\mathtt{equivToEquiv} : (f : A \to B)(e : \mathtt{Equiv}\, f) \to (f' : A \to B) \times (e' : \mathtt{Equiv}\, f') \tag{38}$$

$$\mathtt{equivToEquiv}\, f\, e \triangleq \mathtt{pathToEquiv}\, (\mathtt{fst}(\mathtt{equivToPath}\ f\, e))\, (\mathtt{snd}(\mathtt{equivToPath}\ f\, e))$$

If A and B are fibrant objects, then there are functions

$$\mathtt{pathInv} : (P : A \sim_{\mathcal{U}} B)(\rho : \mathtt{Fib}\, P)(i : \mathtt{I}) \to P\, i \sim_{\mathcal{U}} \mathtt{fst}(\mathtt{pathToPath}\, P\, \rho)\, i \tag{39}$$

$$\mathtt{equivInv} : (f : A \to B)(e : \mathtt{Equiv}\, f) \to f \sim \mathtt{fst}(\mathtt{equivToEquiv}\, f\, e) \tag{40}$$

Since for a function f between fibrant objects $\mathtt{Equiv}\, f$ is a mere proposition [33, Section 3.3], it follows that there is a function:

$$\mathtt{equivInv}' : (f : A \to B)(e : \mathtt{Equiv}\, f) \to (f, e) \sim (\mathtt{equivToEquiv}\, f\, e). \tag{41}$$

Note that for any family $A : \Gamma \to \mathcal{U}$, the type $\mathtt{Fib}\, A$ is also a mere proposition. However, without the presence of a univalent universe it is not possible to state the equivalent of $\mathtt{equivInv}'$ for $\mathtt{pathInv}$. We hope to resolve this issue in subsequent work.

6 Satisfying the Axioms

Working informally in a constructive set theory, the authors of [11] give a model of their type theory using the topos $\hat{\mathcal{C}}$ of contravariant set-valued functors on a particular small category $\mathcal{C}$ that they call the *category of cubes*. Its objects are all finite subsets of a fixed, countably infinite set with decidable equality whose elements should be thought of as names of cartesian directions. Given two such subsets X and Y, the $\mathcal{C}$-morphisms $X \to Y$ are all de Morgan algebra [5, Chapter XI] homomorphisms from the free de Morgan algebra on the finite set Y to that on X; composition and identities are as in the (opposite of the) category of sets. Thus $\mathcal{C}$ is in fact a presentation of the algebraic theory of de Morgan algebra as a Lawvere theory [25, 2] and $\mathcal{C}$ is universal among categories with finite products containing an internal de Morgan algebra.

Let us replace $\mathcal{C}$ by an arbitrary small category $\mathbf{C}$ and see what is required of $\mathbf{C}$ for the topos $\hat{\mathbf{C}}$ of presheaves (within Intuitionistic ZF set theory [1, Section 3.2], say) to have an interval object and subobject of cofibrant propositions satisfying the axioms in Figure 1. We do not aim for complete generality, just enough to encompass some examples of independent interest such as $\mathbf{C} = \mathbf{\Delta}$ the category of inhabited finite linearly ordered sets $[0 < 1 < \cdots < n]$, for which $\hat{\mathbf{C}} = \mathbf{sSet}$, the category of *simplicial sets*, widely used in homotopy theory [18].

6.1 The interval object

We take the interval object $\mathtt{I} \in \hat{\mathbf{C}}$ to be the representable functor $\mathrm{y}_{\mathtt{i}} \triangleq \mathbf{C}(_, \mathtt{i})$ on some object $\mathtt{i} \in \mathbf{C}$. The following theorem gives a useful criterion for such an interval object to satisfy axiom $\mathtt{ax}_1$.

▶ **Theorem 6.1.** *In a presheaf topos $\hat{\mathbf{C}}$, a representable functor $\mathtt{I} = \mathrm{y}_{\mathtt{i}}$ satisfies axiom $\mathtt{ax}_1$ if $\mathbf{C}$ is a* cosifted *category, that is, if finite products in* **Set** *commute with colimits over $\mathbf{C}^{\mathrm{op}}$ [16].*

Proof. $\mathbf{C}$ is cosifted if the colimit functor $\mathrm{colim}_{\mathbf{C}^{\mathrm{op}}} : \hat{\mathbf{C}} \to \mathbf{Set}$ preserves finite products. Recall that $\mathrm{colim}_{\mathbf{C}^{\mathrm{op}}} : \hat{\mathbf{C}} \to \mathbf{Set}$ is left adjoint to the constant presheaf functor $\Delta : \mathbf{Set} \to \hat{\mathbf{C}}$ and (hence) that for any $c \in \mathbf{C}$ it is the case that $\mathrm{colim}_{\mathbf{C}^{\mathrm{op}}} \mathrm{y}_c \cong 1$. So when $\mathbf{C}$ is cosifted we have for any $c \in \mathbf{C}$

$$\hat{\mathbf{C}}(\mathrm{y}_c \times \mathrm{y}_{\mathtt{i}}, \Delta\{0,1\}) \cong \mathbf{Set}(\mathrm{colim}_{\mathbf{C}^{\mathrm{op}}}(\mathrm{y}_c \times \mathrm{y}_{\mathtt{i}}), \{0,1\}) \cong$$
$$\mathbf{Set}(\mathrm{colim}_{\mathbf{C}^{\mathrm{op}}} \mathrm{y}_c \times \mathrm{colim}_{\mathbf{C}^{\mathrm{op}}} \mathrm{y}_{\mathtt{i}}, \{0,1\}) \cong \mathbf{Set}(1 \times 1, \{0,1\}) \cong \{0,1\}$$

Since decidable subobjects in $\hat{\mathbf{C}}$ are classified by $1 + 1 = \Delta\{0,1\}$, this means that the only two decidable subobjects of $\mathrm{y}_c \times \mathrm{y}_{\mathtt{i}}$ are the smallest and the greatest subobjects. Since this is so for all $c \in \mathbf{C}$, it follows that $\mathtt{I} = \mathrm{y}_{\mathtt{i}}$ satisfies $\mathtt{ax}_1$. ◀

A more elementary characterisation of cosiftedness is that $\mathbf{C}$ is inhabited and for every pair of objects $c, c' \in \mathbf{C}$ the category of spans $c \leftarrow \cdot \rightarrow c'$ is a connected category [2, Theorem 2.15]. The category $\mathbf{\Delta}$ has this property and hence the natural candidate for an interval in **sSet**, namely $\mathrm{y}_{\mathtt{i}}$ when $\mathtt{i}$ is the 1-simplex $[0 < 1]$, satisfies $\mathtt{ax}_1$. Any category with finite products trivially has the property. This is the case for $\mathcal{C}$ and thus the interval in the model of [11], where $\mathbf{C} = \mathcal{C}$ and $\mathtt{i}$ is any one-element set (the underlying object of the internal de Morgan algebra), satisfies $\mathtt{ax}_1$.

In addition to $\mathtt{ax}_1$, the other axioms in Figure 1 concerning the interval say that $\mathtt{I}$ is a non-trivial ($\mathtt{ax}_2$) model of the algebraic theory given by $\mathtt{ax}_3$ and $\mathtt{ax}_4$, which we call *path connection algebra*. (See also Definition 1.7 of [17], which considers a similar notion in a more abstract setting.) The 1-simplex in **sSet** is a non-trivial path-connection algebra, the constants being its two end points and the binary operations being induced by the order-preserving binary operations of minimum and maximum on $[0 < 1]$. The generic de Morgan algebra in $\hat{\mathcal{C}}$ is a non-trivial path-connection algebra: the constants are the least and greatest elements and the binary operations are meet and join. An obvious variation on the theme of [11] would be to replace $\mathcal{C}$ by the Lawvere theory for path-connection algebras.

6.2 Cofibrant propositions and the strictness postulate

In a topos with an interval object, there are many candidates for a subobject $\mathtt{Cof} \rightarrowtail \Omega$ satisfying axioms $\mathtt{ax}_5$–$\mathtt{ax}_7$ in Figure 1. For example, one can take the subobject inductively defined by the requirements that it contains $\{\mathtt{0}\}$ and $\{\mathtt{1}\}$ and is closed under binary union and dependent intersection to obtain a smallest $\mathtt{Cof}$ satisfying $\mathtt{ax}_5$–$\mathtt{ax}_7$. Although it is not described that way, this is the notion of cofibrant proposition in $\hat{\mathcal{C}}$ used for the model in [11].

Next we discuss satisfaction of the strictness axiom $\mathtt{ax}_8$ in a general presheaf topos $\hat{\mathbf{C}}$. We work in the CwF associated with $\hat{\mathbf{C}}$ as in [19, Section 4]. In particular, families over a presheaf $\Gamma \in \hat{\mathbf{C}}$ are given by functors $(\int \Gamma)^{\mathrm{op}} \to \mathbf{Set}$, where $\int \Gamma$ is the usual category of elements of Γ. If $\mathcal{S}$ is a Grothendieck universe in the ambient set theory, then its Hofmann-Streicher

lifting [20] to a universe $\mathcal{U}$ in that CwF satisfies that the morphisms $\Gamma \to \mathcal{U}$ in $\hat{\mathbf{C}}$ name the families $(\int \Gamma)^{\mathrm{op}} \to \mathcal{S}$ taking values in $\mathcal{S} \subseteq \mathbf{Set}$.

▶ **Definition 6.2** (Ω_{dec})**.** The subobject classifier Ω in a presheaf topos $\hat{\mathbf{C}}$ maps each $c \in \mathbf{C}$ to the set of *sieves* on c, that is, pre-composition closed subsets $S \subseteq \mathrm{obj}(\mathbf{C}/c)$. Let $\Omega_{\mathrm{dec}} \rightarrowtail \Omega$ be the subpresheaf consisting of those S that are decidable subsets of $\mathrm{obj}(\mathbf{C}/c)$. Of course if the ambient set theory satisfies the Law of Excluded Middle, then $\Omega_{\mathrm{dec}} = \Omega$. In general Ω_{dec} classifies monomorphisms $\alpha : F \rightarrowtail G$ in $\hat{\mathbf{C}}$ for which each injective function $\alpha_c : F(c) \to G(c)$ has decidable image.

▶ **Theorem 6.3.** *Interpreting the universe $\mathcal{U}$ as the Hofmann-Streicher lifting [20] of a Grothendieck universe in* **Set***, a subobject* $\mathtt{Cof} \rightarrowtail \Omega$ *in a presheaf topos $\hat{\mathbf{C}}$ satisfies the strictness axiom* $\mathtt{ax_8}$ *if it is contained in* $\Omega_{\mathrm{dec}} \rightarrowtail \Omega$*.*

Proof. For each $c \in \mathrm{obj}\,\mathbf{C}$, suppose we are given $S \in \Omega_{\mathrm{dec}}(c)$. Thus S is a sieve on c and for each $c' \in \mathrm{obj}\,\mathbf{C}$ and $\mathbf{C}$-morphism $f : c' \to c$, it is decidable whether or not $f \in S$. We can also regard S as a subpresheaf $S \hookrightarrow \mathrm{y}_c$.

Suppose that we have families $A : (\int S)^{\mathrm{op}} \to \mathcal{S}$, $B : (\int \mathrm{y}_c)^{\mathrm{op}} \to \mathcal{S}$ and a natural isomorphism s between A and the restriction of B along $S \hookrightarrow \mathrm{y}_c$. For each $\mathbf{C}$-morphism $\cdot \xrightarrow{f} c$, using the decidability of S, we can define bijections $s'(f) : B'(f) \cong B(f)$ given by

$$B'(f) \triangleq \begin{cases} A(f) & \text{if } f \in S \\ B(f) & \text{if } \neg(f \in S) \end{cases} \qquad \text{and} \qquad s'(f) \triangleq \begin{cases} s(f) & \text{if } f \in S \\ f & \text{if } \neg(f \in S) \end{cases}$$

(compare this with Definition 15 in [11]). We make B' into a functor $(\int \mathrm{y}_c)^{\mathrm{op}} \to \mathcal{S}$ by transferring the functorial action of B across these bijections. Having done that, s' becomes a natural isomorphism $B' \cong B$; and by definition its restriction along $S \hookrightarrow \mathrm{y}_c$ is s. ◀

▶ Remark 6.4. As a partial converse of the theorem, we have that if $\mathtt{ax_8}$ is satisfied by the Hofmann-Streicher universe in the CwF associated with $\hat{\mathbf{C}}$, then each cofibrant mono $\alpha : F \rightarrowtail G$ has component functions $\alpha_c : F\,c \to G\,c$ whose images are $\neg\neg$-closed subsets of $G\,c$. To see this we can apply an argument due to Andrew Swan [private communication] that relies upon the fact that in the ambient set theory one has

$$(X = \emptyset) = \forall x \in X.\ \bot = \neg\neg(\forall x \in X.\ \bot) = \neg\neg(X = \emptyset) \tag{42}$$

For suppose given $c \in \mathrm{obj}\,\mathbf{C}$ and $S \in \mathtt{Cof}(c)$. We have to use axiom $\mathtt{ax_8}$ to show that S is a $\neg\neg$-closed subset of $\mathrm{obj}(\mathbf{C}/c)$. Let $A : (\int S)^{\mathrm{op}} \to \mathcal{S}$ be the constant functor mapping each (c', f) to $\{\emptyset\}$; and let $B : (\int \mathrm{y}_c)^{\mathrm{op}} \to \mathcal{S}$ map each (c', f) to $\{\{\emptyset\}, \{\emptyset \mid f \in S\}\}$ (which does extend to a functor, because S is a sieve). The restriction of B along $S \hookrightarrow \mathrm{y}_c$ is isomorphic to A and so by $\mathtt{ax_8}$ there some $B' : (\int \mathrm{y}_c)^{\mathrm{op}} \to \mathcal{S}$ whose restriction along $S \hookrightarrow \mathrm{y}_c$ is equal to A and some isomorphism $s' : B' \cong B$. For any $(c', f) \in \mathrm{obj}(\int \mathrm{y}_c)$, suppose $X \in B'(c', f)$; then $f \in S \Rightarrow X = \emptyset$, hence $\neg\neg(f \in S) \Rightarrow \neg\neg(X = \emptyset)$ and therefore by (42), $\neg\neg(f \in S) \Rightarrow (X = \emptyset)$. Therefore $\neg\neg(f \in S) \Rightarrow B'(c', f) = \{\emptyset\} \Rightarrow B(c', f) \cong \{\emptyset\} \Rightarrow f \in S$. So S is indeed a $\neg\neg$-closed subset of $\mathrm{obj}(\mathbf{C}/c)$.

Note that this result implies that it is not possible to take $\mathtt{Cof}$ to be the whole of Ω and satisfy $\mathtt{ax_8}$ unless the ambient set theory satisfies the Law of Excluded Middle.

It is not hard to see that Ω_{dec} satisfies $\mathtt{ax_6}$ and $\mathtt{ax_7}$. It also satisfies $\mathtt{ax_5}$ if for example equality of $\mathbf{C}$-morphisms is decidable. Therefore we get a model of our axioms in $\hat{\mathbf{C}}$ by taking $\mathtt{Cof} = \Omega_{\mathrm{dec}}$, for any cosifted $\mathbf{C}$ that has decidable equality of morphisms and a representable

with the structure of a path-connection algebra; $\mathcal{C}$ and $\mathbf{\Delta}$ are both examples. In the case of $\hat{\mathcal{C}}$ this is a different choice of cofibrant proposition to the one in [11]. In the case of $\hat{\mathbf{\Delta}} = \mathbf{sSet}$, it remains to be investigated what is the relationship between this constructive model based on CCHM fibrations and Voevodsky's non-constructive model of univalent type theory using classical Kan simplicial sets [23].

7 Related and Future Work

The work reported here was inspired by [11]. We have shown how to express Cohen, Coquand, Huber and Mörtberg's notion of fibration in the internal type theory of a topos (Definition 4.3). We found that quite a simple collection of axioms (Figure 1) suffices for this to model Martin-Löf type theory with path types satisfying a weak form of univalence. The construction in Section 8.2 of [11] of a fibrant universe satisfying the full univalence axiom uses a modified form of Hofmann-Streicher lifting [20] within the ambient constructive set theory. We plan to investigate whether that, or indeed some other universe construction, can be axiomatised within the internal type theory of a topos.

We found that only a simple path-connection algebra, rather than a de Morgan algebra structure, is needed on the interval. Furthermore, the collection of propositions suitable for uniform Kan filling can be chosen in various ways. This allows a model of our axioms in constructive simplicial sets as one example, besides variations on the notion of cubical set. In Section 6 we only considered how presheaf categories can satisfy our axioms. It might be interesting to consider models in sheaf toposes, particularly *gros toposes* such as [22] that allow the interval object to be the usual topological interval.

Our concerns (modelling intensional type theory with path-like identity types, getting cubical and simplicial sets as instances) are similar to those of Gambino and Sattler [17]; although our methods differ, it seems we arrive at the same notion of fibration, although this remains to be investigated. Birkedal *et al* [8] are developing guarded cubical type theory with a semantics based on an axiomatic version of [11] within the internal logic of a presheaf topos. We believe that our approach to modelling cubical type theory is more general than theirs, but that there will be interesting synergies.

Acknowledgements. We thank Thierry Coquand for many conversations about the results described in [11] and the possibility of an internal characterisation of its constructions. We are grateful to him, Martín Escardó, Anders Mörtberg, Christian Sattler, Bas Spitters and Andrew Swan for comments on the work reported here.

References

1 P. Aczel and M. Rathjen. Notes on constructive set theory. Book draft, 2010.

2 J. Adámek, J. Rosický, and E. M. Vitale. *Algebraic Theories: a Categorical Introduction to General Algebra*, volume 184 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 2010.

3 Agda Project. URL: `http://wiki.portal.chalmers.se/agda`.

4 S. Awodey. Natural models of homotopy type theory. *ArXiv e-prints*, arXiv:1406.3219v2 [math.CT], March 2015.

5 R. Balbes and P. Dwinger. *Distributive Lattices*. University of Missouri Press, 1975.

6 M. Bezem and T. Coquand. A Kripke model for simplicial sets. *Theoretical Computer Science*, 574:86–91, 2015.

7 M. Bezem, T. Coquand, and S. Huber. A model of type theory in cubical sets. In R. Matthes and A. Schubert, editors, *Proc. TYPES 2013*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 107–128. Leibniz-Zentrum fuer Informatik, 2014.

8 L. Birkedal, A. Bizjak, R. Clouston, H. B. Grathwohl, B. Spitters, and A. Verozzi. Guarded cubical type theory. Abstract of a talk at TYPES 2016, Novi Sad, Serbia, 2016.

9 R. Brown and G. H. Mosa. Double categories, 2-categories, thin structures and connections. *Theory and Applications of Categories*, 5(7):163–175, 1999.

10 J. Cockx and A. Abel. Sprinkles of extensionality for your vanilla type theory. Abstract of a talk at TYPES 2016, Novi Sad, Serbia, 2016.

11 C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. Preprint, December 2015.

12 T. Coquand. A cubical type theory, August 2015. Slides of an invited talk at Domains XII, University of Cork, Ireland, `www.cse.chalmers.se/~coquand/cork.pdf`.

13 T. Coquand and N. A. Danielsson. Isomorphism is equality. *Indagationes Mathematicae*, 24(4):1105–1120, 2013.

14 P. Dybjer. Internal type theory. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer Berlin Heidelberg, 1996.

15 M. Escardo. A small type of propositions *à la* Voevodsky in Agda, August 2015. URL: `http://www.cs.bham.ac.uk/~mhe/impredicativity/`.

16 P. Gabriel and F. Ulmer. *Lokal Präsentierbare Kategorien*, volume 221 of *Lecture Notes in Mathematics*. Springer-Verlag, 1971.

17 N. Gambino and C. Sattler. Uniform Fibrations and the Frobenius Condition. *ArXiv e-prints*, arXiv:1510.00669, October 2015.

18 P. G. Goerss and J. F. Jardine. *Simplicial Homotopy Theory*. Modern Birkhäuser Classics. Birkhäuser Basel, 2009.

19 M. Hofmann. Syntax and semantics of dependent types. In A. M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, pages 79–130. Cambridge University Press, 1997.

20 M. Hofmann and T. Streicher. Lifting Grothendieck universes. Unpublished note, 1999.

21 S. Huber. *A Model of Type Theory in Cubical Sets*. Licentiate thesis, University of Gothenburg, May 2015.

22 P. T. Johnstone. On a topological topos. *Proceedings of the London Mathematical Society*, s3-38(2):237–271, 1979.

23 C. Kapulkin, P. L. Lumsdaine, and V. Voevodsky. The simplicial model of univalent foundations. *arXiv preprint*, arXiv:1211.2851v2 [math.LO], April 2014.

24 J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

25 F. W. Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(1):869–872, 1963.

26 P. L. Lumsdaine and M. A. Warren. The local universes model: An overlooked coherence construction for dependent type theories. *ACM Trans. Comput. Logic*, 16(3):23:1–23:31, July 2015.

27 M. E. Maietti. Modular correspondence between dependent type theories and categories including pretopoi and topoi. *Mathematical Structures in Computer Science*, 15:1089–1149, 2005.

28 P. J. May. *A Concise Course in Algebraic Topology*. Chicago Lectures in Mathematics. University of Chicago Press, 1999.

29 A. 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*, pages 40–128. Oxford University Press, 2000.

30 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013.

31 B. Spitters. Cubical sets as a classifying topos. Abstract of a talk at TYPES 2015, Tallinn, Estonia, 2015.

32 A. Swan. An algebraic weak factorisation system on 01-substitution sets: A constructive proof. *ArXiv e-prints*, arXiv:1409.1829, September 2014. URL: `http://arxiv.org/abs/1409.1829`.

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

34 B. van den Berg. Path categories and propositional identity types. *ArXiv e-prints*, arXiv:1604.06001 [math.CT], April 2016.

Bar Recursion in Classical Realisability: Dependent Choice and Continuum Hypothesis

Jean-Louis Krivine

University Paris-Diderot, CNRS, France
krivine@pps.univ-paris-diderot.fr

Abstract

This paper is about the bar recursion operator in the context of classical realizability. The pioneering work of Berardi, Bezem, Coquand [1] was enhanced by Berger and Oliva [2]. Then Streicher [12] has shown, by means of their bar recursion operator, that the realizability models of ZF, obtained from usual models of λ-calculus (Scott domains, coherent spaces, ...), satisfy the axiom of dependent choice. We give a proof of this result, using the tools of classical realizability. Moreover, we show that these realizability models satisfy the well ordering of $\mathbb{R}$ and the continuum hypothesis. These formulas are therefore realized by closed λ_c-terms. This new result allows to obtain programs from proofs of arithmetical formulas using all these axioms.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases lambda-calculus, Curry-Howard correspondence, set theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.25

1 Introduction

This paper is about the bar recursion operator [11], in the context of classical realizability [8, 9]. It is a sequel to the three papers [1, 2, 12]. We use the definitions and notations of the theory of classical realizability as expounded in [7, 8, 9].

In [1], S. Berardi, M. Bezem and T. Coquand have shown that a form of the bar recursion operator can be used, in a proof-program correspondence, to interpret *the axiom of dependent choice* in proofs of Π^0_2-formulas of arithmetic. Their work was enhanced and adapted to the theory of domains by U. Berger and P. Oliva in [2]. In [12], T. Streicher has shown, by using the bar recursion operator of [2], that the models of ZF, associated with realizability algebras [7, 9] obtained from usual models of λ-calculus (Scott domains, coherent spaces, ...), satisfy the axiom of dependent choice. We give here a proof of this result, but for a realizability algebra which is built following the presentation of [1], which we call the BBC-algebra.

In Section 2, we define and study this algebra; we define also the bar recusion operator, which is a closed λ-term. In Sections 3 and 4, which are very similar, we show that this operator realizes the axiom of countable choice (CC), then the axiom of dependent choix (DC). The proof is a little simpler for CC. In Section 5, we deduce from this result, using results of [10] that, in the model of ZF associated with this realizability algebra, *every real (more generally, every sequence of ordinals) is constructible*. The formulas "$\mathbb{R}$ *is well ordered*" and "*Continuum Hypothesis*" are therefore realized in these models by a closed λ_c-term (i.e. a λ-term containing the control instruction cc of Felleisen-Griffin). We show also that *every true formula of analysis is realized by a closed* λ_c*-term*. Using these new results, we show how to obtain a program (closed λ_c-term) from any proof of an arithmetical formula in the theory ZF + "Dependent choice" + "Every real is constructible" (and therefore "Well ordering of $\mathbb{R}$" and "Continuum Hypothesis").

© Jean-Louis Krivine;
licensed under Creative Commons License CC-BY

Remark. In [8], we obtain programs from proofs in ZF + "Dependent choice", using another realizability algebra (the *thread model*). We cannot use Continuum Hypothesis in this context but, on the other hand, we are not limited to proofs of arithmetical formulas.

2 The BBC realizability algebra

The definition and general properties of a *realizability algebra* are given at the beginning of [7]. It consists of a set of *terms* Λ, a set of *stacks* Π and a set of *processes* $\Lambda \star \Pi$. Every closed λ-term is interpreted as a term.

In this paper, we consider a particular realizability algebra $\boldsymbol{\mathcal{B}} = (\mathbf{\Lambda}, \mathbf{\Pi}, \perp\!\!\!\perp)$, which we call the *BBC algebra* because it is a reformulation of the programming language of [1]. It is defined as follows:

- The set of *processes* $\mathbf{\Lambda} \star \mathbf{\Pi}$ is $\mathbf{\Lambda} \times \mathbf{\Pi}$.
- The set of *terms* $\mathbf{\Lambda}$ is the smallest set which contains the following *constants of term*: B, C, I, K, W *(Curry's combinators)*, cc *(Felleisen-Griffin instruction)*, A *(abort instruction)*, $\mathsf{p}, \mathsf{q}_0, \dots, \mathsf{q}_N$ *(variables)* where N is a fixed integer; and is such that:

  if $\xi, \eta \in \mathbf{\Lambda}$ then $(\xi)\eta \in \mathbf{\Lambda}$ *(application)* ;

  with each sequence $\xi_i (i \in \mathbb{N})$ of closed elements of $\mathbf{\Lambda}$ (i.e. which contain no variable $\mathsf{p}, \mathsf{q}_0, \dots, \mathsf{q}_N$) is associated, in a one-to-one (and well founded) way, a constant of term denoted by $\bigwedge_i \xi_i$.
  Therefore, each term $\xi \in \mathbf{\Lambda}$ is a finite sequence of constants of term and parentheses.
  $\mathbf{\Lambda}$ is defined by an induction of length $\aleph_1$ and is of cardinality $2^{\aleph_0}$.

Notations. The application $(\dots((\xi_1)\xi_2)\dots)\xi_n$ will be often written $(\xi_1)\xi_2\dots\xi_n$ or even $\xi_1\xi_2\dots\xi_n$. The finite sequence $\mathsf{q}_0, \dots, \mathsf{q}_N$ will be often written $\vec{\mathsf{q}}$.

- The set of *stacks* $\mathbf{\Pi}$ is defined as follows: a stack π is a finite sequence $t_0 \cdot \dots \cdot t_{n-1} \cdot \pi_0$ with $t_0, \dots, t_{n-1} \in \mathbf{\Lambda}$; it is terminated by the symbol π_0 which represents the *empty stack*.

For each stack π, the *continuation* k_π is a term which is defined by recurrence: $\mathsf{k}_{\pi_0} = \mathsf{A}$; $\mathsf{k}_{t \cdot \pi} = \ell_t \, \mathsf{k}_\pi$, with $\ell_t = ((\mathsf{C})(\mathsf{B})\mathsf{CB})t$ or $\lambda k \lambda x (k)(x)t$.

Thus, if the stack π is $t_0 \cdot \dots \cdot t_{n-1} \cdot \pi_0$, we have:

$$\mathsf{k}_\pi = (\ell_{t_0}) \dots (\ell_{t_{n-1}}) \mathsf{A} or \lambda x (\mathsf{A})(x) t_0 \dots t_{n-1} \, .$$

The *integer* $\underline{n}$ is defined as follows:

$$\underline{0} = (\mathsf{K})\mathsf{I} or \lambda x \lambda y \, y ; \; \underline{n+1} = (\sigma)\underline{n} with \sigma = (\mathsf{BW})(\mathsf{C})(\mathsf{B})\mathsf{BB} or \lambda n \lambda f \lambda x (f)(n) f x \, .$$

The relation of *execution* $\succ$ is the least preorder on $\mathbf{\Lambda} \star \mathbf{\Pi}$ defined by the following rules (with $\xi, \eta, \zeta \in \mathbf{\Lambda}, \pi \in \mathbf{\Pi}$ and $n \in \mathbb{N}$):

1. $(\xi)\eta \star \pi \succ \xi \star \eta \cdot \pi$; *(push)*
2. $\mathsf{B} \star \xi \cdot \eta \cdot \zeta \cdot \pi \succ \xi \star (\eta)\zeta \cdot \pi$; *(apply)*
3. $\mathsf{C} \star \xi \cdot \eta \cdot \zeta \cdot \pi \succ \xi \star \zeta \cdot \eta \cdot \pi$; *(switch)*
4. $\mathsf{I} \star \xi \cdot \pi \succ \xi \star \pi$; *(no operation)*
5. $\mathsf{K} \star \xi \cdot \eta \cdot \pi \succ \xi \star \pi$; *(delete)*
6. $\mathsf{W} \star \xi \cdot \eta \cdot \pi \succ \xi \star \eta \cdot \eta \cdot \pi$; *(copy)*
7. $\mathsf{cc} \star \xi \cdot \pi \succ \xi \star \mathsf{k}_\pi \cdot \pi$; *(save the stack)*
8. $\mathsf{A} \star \xi \cdot \pi \succ \xi \star \pi_0$; *(abort)* or *(delete the stack)*
9. $\bigwedge_i \xi_i \star \underline{n} \cdot \pi \succ \xi_n \star \pi$; *(oracle)*

There is no execution rule for p and q_i which are, therefore, *stop instructions*; p plays a special role because of the definition of $\perp\!\!\!\perp$ below.

When $\xi, \eta \in \mathbf{\Lambda}$, we set $\xi \succ \eta$ iff $(\forall \pi \in \mathbf{\Pi})(\xi \star \pi \succ \eta \star \pi)$.

- *Execution of processes.* For every process $\xi \star \pi$, at most one among the rules 1 to 9 applies. By iterating these rules, we obtain the *reduction* or the *execution* of the process $\xi \star \pi$. This execution stops if and only if the stack is insufficient (rules 2 to 8) or does not begin with an integer (rule 9) or else if the process has the form $\mathsf{p} \star \varpi$ or $\mathsf{q}_i \star \varpi$.
- *Definition of* $\perp\!\!\!\perp$. We set $\perp\!\!\!\perp = \{\xi \star \pi \in \mathbf{\Lambda} \star \mathbf{\Pi}\ ;\ (\exists \varpi \in \mathbf{\Pi})(\xi \star \pi \succ \mathsf{p} \star \varpi)\}$.
- *Proof-like terms.* Let PL_0 be the countable set of terms built with the constants B, C, I, K, W, cc and the application. It is the smallest possible set of *proof-like terms*. We shall also consider the set PL of *closed terms* (i.e. with no occurrence of $\mathsf{p}, \vec{\mathsf{q}}$) which is much bigger: it contains A and the oracles $\bigwedge_i \xi_i$ and is therefore of cardinality $2^{\aleph_0}$.

▶ **Lemma 1.** *$\mathcal{B}$ is a coherent realizability algebra.*

Proof. *$\mathcal{B}$ is a realizability algebra*: It remains to check that $\mathsf{k}_\pi \star \xi \cdot \varpi \succ \xi \star \pi$, which is done by recurrence on π:

- if $\pi = \pi_0$, it is rule 8;
- if $\pi = t \cdot \rho$ we have $\mathsf{k}_\pi \star \xi \cdot \varpi = \mathsf{k}_{t \cdot \rho} \star \xi \cdot \varpi = \ell_t \mathsf{k}_\rho \star \xi \cdot \varpi \succ (\mathsf{k}_\rho)(\xi) t \star \varpi \succ \mathsf{k}_\rho \star \xi t \cdot \varpi \succ \xi t \star \rho$ (recurrence hypothesis) $\succ \xi \star t \cdot \rho$.

$\mathcal{B}$ is coherent: If $\theta \in \mathsf{PL}$ then $\theta \star \pi_0 \notin \perp\!\!\!\perp$; indeed, p does not appear during the execution of $\theta \star \pi_0$. ◀

Models and functionals

A coherent realizability algebra is useful in order to give *truth values* to formulas of ZF. In fact, we use a theory called ZF_ε [8] which is a conservative extension of ZF. This theory has an additional strong membership relation symbol ε *which is not extensional*.

For each closed formula F of ZF_ε, we define *two truth values*, denoted $\|F\|$ and $|F|$, with $\|F\| \subset \mathbf{\Pi}$ and $|F| \subset \mathbf{\Lambda}$, with the relation $\xi \in |F| \Leftrightarrow (\forall \pi \in \|F\|)(\xi \star \pi \in \perp\!\!\!\perp)$. The relation $\xi \in |F|$ is also written $\xi \Vdash F$ and reads *"the term ξ realizes the formula F"*. All the necessary definitions are given in [7, 8, 9].

The following Lemma 2 is a useful property of the BBC realizability algebra $\mathcal{B}$.

▶ **Lemma 2.** *For all formulas A, B of ZF_ε, and all terms $\xi \in \mathbf{\Lambda}$, we have:*

$$\xi \Vdash A \to B \text{ iff } (\forall \eta \in \mathbf{\Lambda})(\eta \Vdash A \Rightarrow \xi\eta \Vdash B)\,.$$

Proof. Indeed, by the general definition of $\Vdash$, we have:

$$(\xi \Vdash A \to B) \Leftrightarrow (\forall \eta \Vdash A)(\forall \pi \in \|B\|)(\xi \star \eta \cdot \pi \in \perp\!\!\!\perp)\,.$$

Now, by the above definition of $\perp\!\!\!\perp$, it is clear that $(\xi \star \eta \cdot \pi \in \perp\!\!\!\perp) \Leftrightarrow (\xi\eta \star \pi \in \perp\!\!\!\perp)$ from which the result follows. ◀

Classical realizability is an extension of forcing. As in forcing, we start with an ordinary model $\mathcal{M}$ of ZFC or even ZF + V = L (the Gödel constructibility axiom) which we call the *ground model*, and we build a *realizability model* $\mathcal{N}$ which satisfies ZF_ε in the following sense: $\mathcal{M}$ and $\mathcal{N}$ have the same domain, but neither the same language, nor the same truth values. The language of $\mathcal{N}$ has the additional binary symbol ε of *strong membership*. The truth values of $\mathcal{N}$ are not $0, 1$ as for $\mathcal{M}$, but are taken in $\mathcal{P}(\mathbf{\Pi})$ endowed with a suitable structure

of Boolean algebra [7, 9]. We say that $\mathcal{N}$ *satisfies a formula* F iff there is a proof-like term θ which realizes F or equivalently, if the truth value $\|F\|$ of F is the unit of the Boolean algebra $\mathcal{P}(\mathbf{\Pi})$.

A *functional* on the ground model $\mathcal{M}$ is a formula $F(\vec{x}, y)$ of ZF with parameters in $\mathcal{M}$, such that $\mathcal{M} \models \forall \vec{x}\, \exists! y\, F(\vec{x}, y)$. Denoting such a functional by f, we write $y = f(\vec{x})$ for $F(\vec{x}, y)$.

Since $\mathcal{M}$ and $\mathcal{N}$ have the same domain, all the functionals defined on $\mathcal{M}$ are also defined on $\mathcal{N}$ and *they satisfy the same equations* and even the same *Horn formulas* i.e. formulas of the form $\forall \vec{x}(f_1(\vec{x}) = g_1(\vec{x}), \dots, f_n(\vec{x}) = g_n(\vec{x}) \to f(\vec{x}) = g(\vec{x}))$.

A particularly useful binary functional on $\mathcal{M}$ (and thus also on $\mathcal{N}$) is the *application*, denoted by app, which is defined as follows: $\mathrm{app}(f, x) = \{y\ ;\ (x, y) \in f\}$.

We shall often write $f[x]$ for $\mathrm{app}(f, x)$. This allows to consider each set in $\mathcal{M}$ (and in $\mathcal{N}$) as a unary functional.

▶ Remark. We can define a set f in $\mathcal{M}$ by giving $f[x]$ for every x, provided that there exists a set X such that $f[x] = \emptyset$ for all $x \notin X$: take $f = \bigcup_{x \in X} \{x\} \times f[x]$.

In the ground model $\mathcal{M}$, every function is defined in this way but in general, *this is false in* $\mathcal{N}$.

Quantifiers restricted to $\mathbb{N}$

In [9], we defined the quantifier $\forall x^{\mathrm{int}}$, by setting:

$$\|\forall x^{\mathrm{int}} F[x]\| = \bigcup_{n \in \mathbb{N}} \|\{\underline{n}\} \to F[n]\} = \{\underline{n} \cdot \pi\ ;\ n \in \mathbb{N}, \pi \in \|F[n]\|\}\,,$$

so that we have:

$$\xi \Vdash \forall x^{\mathrm{int}} F[x] \ \Leftrightarrow\ \xi \underline{n} \Vdash F[n] \text{ for all } n \in \mathbb{N}\,;$$

and it is shown that it is a correct definition of the restricted quantifier to $\mathbb{N}$.

Indeed the equivalence $\forall x^{\mathrm{int}} F[x] \leftrightarrow \forall x(\mathrm{int}[x] \to F[x])$ is realized by a closed λ-term independent of F, called a *storage operator*.

The formula $\mathrm{int}[x]$ is any formula of ZF which says that x is an integer.

▶ **Theorem 3.** *If we take* PL *for the set of proof-like terms, and if the ground model* $\mathcal{M}$ *is transitive and countable, then there exists a countable realizability model* $\mathcal{N}$ *which has only standard integers, i.e. which is an* ω*-model.*

Proof. Let $\mathcal{T}$ be the theory formed with closed formulas, with parameters in $\mathcal{M}$, which are realized by a proof-like term. $\mathcal{T}$ is ω-complete: indeed, if $\theta_n \in \mathsf{PL}$ and $\theta_n \Vdash F[n]$ for $n \in \mathbb{N}$, let us set $c = \bigwedge_i \theta_i$. Then $c\underline{n} \Vdash F[n]$ for all $n \in \mathbb{N}$ and therefore $c \Vdash \forall n^{\mathrm{int}} F[n]$, i.e. $\forall n^{\mathrm{int}} F[n] \in \mathcal{T}$. It follows that $\mathcal{T}$ has a countable ω-model. ◀

▶ **Proposition 4.** *Let* $f : \mathbb{N} \to 2$ *and* $\theta \in$ *PL*, $\theta \Vdash \exists n^{int}(f(n) = 1)$. *Then* $\theta \star p \cdot \pi_0 \succ p \star \underline{n} \cdot \varpi$ *with* $f(n) = 1$.

Proof. There exists $\tau \in \mathbf{\Lambda}$ such that $\tau\underline{n} \succ \mathsf{p}$ if $f(n) = 1$ and $\tau\underline{n} \succ \mathsf{q}_0$ if $f(n) = 0$: set $\tau = \lambda x(\bigwedge_i \xi_i) x\, \mathsf{pq}_0$ with $\xi_n = \mathsf{K}$ if $f(n) = 1$ and $\xi_n = \mathsf{KI}$ if $f(n) = 0$.

Then we have $\tau \Vdash \forall n^{\mathrm{int}}(f(n) \neq 1)$ and therefore $\theta\tau \Vdash \bot$. We necessarily have: $\theta \star \tau \cdot \pi_0 \succ \tau \star \underline{n} \cdot \pi$ for some n; furthermore, we have $\tau\underline{n} \succ \mathsf{p}$, otherwise we should have $\tau\underline{n} \succ \mathsf{q}_0$, and thus $\theta \star \tau \cdot \pi_0 \notin \perp\!\!\!\perp$. Therefore $f(n) = 1$. ◀

▶ Remark. This shows that, from any proof-like term which realizes a given Σ^0_1 arithmetical formula, we obtain a program which computes an integer satisfying this formula. Such a realizer is given by any proof of this formula by means of axioms which have themselves such realizers.

The theory of classical realizability gives realizers for the axioms of ZF. We show below that the bar recursion operator realizes the axiom of dependent choice. Finally, in Section 5, we get (rather complicated) proof-like realizers for the axioms "$\mathbb{R}$ is well ordered" and "Continuum hypothesis".

Execution of processes

Notation. If $\pi = t_0 \cdot \ldots \cdot t_{n-1} \cdot \pi_0$, we shall write $\pi \cdot t$ for $t_0 \cdot \ldots \cdot t_{n-1} \cdot t \cdot \pi_0$. Thus, we obtain $\mathsf{k}_{\pi \cdot t}$ by replacing, in k_π, the last occurrence of A by $\ell_t \mathsf{A}$.

▶ **Lemma 5.** *If $\xi \star \pi \in \perp\!\!\!\perp$, then $\xi' \star \pi' \in \perp\!\!\!\perp$ and $\xi' \star \pi' \cdot t \in \perp\!\!\!\perp$, where $\xi' \star \pi'$ is obtained by replacing, in $\xi \star \pi$, some occurrences of A by $(\ell_u)\mathsf{A} = \mathsf{k}_{u \cdot \pi_0}$ and some occurrences of the variables $\mathsf{q}_0, \ldots, \mathsf{q}_N$ by $t_0, \ldots, t_N$; $t_0, \ldots, t_N, t, u$ are arbitrary terms.*

▶ Remark. In particular, it follows that $\xi \star \pi_0 \in \perp\!\!\!\perp \Rightarrow \xi \Vdash \bot$.

Proof. Proof by recurrence on the length of the execution of $\xi \star \pi \in \perp\!\!\!\perp$ by means of rules 1 to 9. We consider the last used rule. There are two non trivial cases:

- Rule 7 (execution of cc); we must show $\mathsf{cc} \star \xi' \cdot \pi' \cdot t \in \perp\!\!\!\perp$.
  We apply the recurrence hypothesis to $\xi \star \mathsf{k}_\pi \cdot \pi$, in which we replace:
  - π_0 by $t \cdot \pi_0$ (thus π becomes $\pi \cdot t$);
  - the last occurrence of A in $\mathsf{k}_\pi = (\ell_{t_0}) \ldots (\ell_{t_{n-1}})\mathsf{A}$ by $(\ell_t)\mathsf{A}$ (thus k_π becomes $\mathsf{k}_{\pi \cdot t}$).

  Then, we make the substitutions in ξ, π, which gives $\xi' \star \mathsf{k}_{\pi' \cdot t} \cdot \pi' \cdot t$.
- Rule 8 (execution of A); we must show $(\ell_u)\mathsf{A} \star \xi' \cdot \pi' \cdot t \in \perp\!\!\!\perp$.
  We apply the recurrence hypothesis to $\xi \star \pi_0$, which gives $\xi' \star u \cdot \pi_0 \in \perp\!\!\!\perp$, thus $\xi' u \star \pi_0 \in \perp\!\!\!\perp$ and therefore $\mathsf{A} \star \xi' u \cdot \pi' \cdot t \in \perp\!\!\!\perp$ (rule 8); finally, we obtain $(\ell_u)\mathsf{A} \star \xi' \cdot \pi' \cdot t \in \perp\!\!\!\perp$. ◀

In each process $\xi \star \pi \in \perp\!\!\!\perp$, we define an occurrence of p, which is called *efficient*, by recurrence on the length of its reduction. If $\xi = \mathsf{p}$, it is this very occurrence. Otherwise, we consider the first rule used in the reduction, and the definition is clear; for example, if it is rule 7, and if the efficient occurrence in $\xi \star \mathsf{k}_\pi \cdot \pi$ is in k_π or in π, then we take the corresponding occurrence in $\mathsf{cc} \star \xi \cdot \pi$.

▶ **Lemma 6.** *If $\xi \star \pi \in \perp\!\!\!\perp$, then:*

- *$\xi' \star \pi' \in \perp\!\!\!\perp$, where $\xi' \star \pi'$ is obtained by substituting arbitrary terms for the non efficient occurrences of p.*
- *$\xi' \star \pi' \notin \perp\!\!\!\perp$ and indeed $\xi' \star \pi' \succ \mathsf{q}_0 \star \varpi$, where $\xi' \star \pi'$ is obtained by substituting q_0 for the efficient occurrence of p, and arbitrary terms for the non efficient occurrences of p.*

Proof. The proof is immediate, by recurrence on the length of the reduction of $\xi \star \pi$ by means of rules 1 to 9: consider the last used rule. ◀

▶ **Corollary 7.** *If $\xi \Vdash \top, \bot \to \bot$ and $\xi \Vdash \bot, \top \to \bot$, then $\xi \Vdash \top, \top \to \bot$, and thus:*

$$\lambda x(x)II \Vdash \neg\forall x^{\mathbb{N}^2}(x \neq 0, x \neq 1 \to \bot)$$

and

$$W \Vdash \forall x^{\mathbb{N}^2}(\forall y^{\mathbb{N}^2}(y \neq 0, y \neq x \to y \not\leq x), x \neq 0 \to \bot)\,.$$

▶ Remark. These two formulas express respectively that the Boolean algebra $\beth 2$, which is defined in [9, 10], is non trivial and even atomless. Intuitively, $\beth 2$ represents the *type of Booleans* of the realizability model $\mathcal{N}$. It is called the *characteristic Boolean algebra* of $\mathcal{N}$.

Proof. We apply Lemma 6 to $\xi \star \mathsf{p} \cdot \mathsf{p} \cdot \pi_0$. We have $\xi \star \mathsf{q}_0 \cdot \mathsf{p} \cdot \pi_0 \in \perp\!\!\!\perp$ and $\xi \star \mathsf{p} \cdot \mathsf{q}_0 \cdot \pi_0 \in \perp\!\!\!\perp$, which shows that the efficient occurrence of p is in ξ. Therefore $\xi \star t \cdot u \cdot \pi_0 \in \perp\!\!\!\perp$ for every $t, u \in \mathbf{\Lambda}$, again by Lemma 6.

The last two assertions follow from the fact that:

$$\|\forall x^{\beth 2}(x \neq 0, x \neq 1 \to \bot)\| = \|\top, \bot \to \bot\| \cup \|\bot, \top \to \bot\|$$

and therefore:

$$|\forall x^{\beth 2}(x \neq 0, x \neq 1 \to \bot)| = |\top, \top \to \bot|\,. \qquad \blacktriangleleft$$

▶ **Theorem 8.** *For every sequence $\xi_i \in \mathbf{\Lambda}$ $(i \in \mathbb{N})$, there exists $\phi \in \mathbf{\Lambda}$ such that:*

- *$\phi \underline{i} \succ \xi_i$ for every $i \in \mathbb{N}$;*
- *for every $U \in \mathbf{\Lambda}$ such that $U\phi \Vdash \bot$, there exists $k \in \mathbb{N}$ such that $U\psi \Vdash \bot$ for every $\psi \in \mathbf{\Lambda}$ such that $\psi \underline{i} \succ \xi_i$ for every $i < k$.*

▶ Remark. Theorem 8 will be used in order to show properties of the bar recursion operator. In fact, the following weaker formulation is sufficient:

For every sequence $\xi_i \in \mathbf{\Lambda}$ $(i \in \mathbb{N})$ and every $U \in \mathbf{\Lambda}$ such that:

$$(\forall k \in \mathbb{N})(\exists \psi \in \mathbf{\Lambda})\{U\psi \nVdash \bot, (\forall i < k)(\psi \underline{i} \succ \xi_i)\}$$

there exists $\phi \in \mathbf{\Lambda}$ such that $U\phi \nVdash \bot$ and $(\forall i \in \mathbb{N})(\phi \underline{i} \succ \xi_i)$.

In the particular case of forcing, this is exactly the *decreasing chain condition*: every decreasing sequence of (non false) conditions has a lower bound (which is non false).

Proof. We set $\eta_i = \lambda \mathsf{p} \lambda \vec{\mathsf{q}}\, \xi_i$; thus, we have $\eta_i \in \mathsf{PL}$ and $\eta_i \mathsf{p} \vec{\mathsf{q}} \succ \xi_i$. Let $\eta = \bigwedge_i \eta_i$ and $\phi = \lambda x(\eta) x\, \mathsf{p} \vec{\mathsf{q}}$. Thus, we have $\eta \in \mathsf{PL}$ and $\phi \underline{i} \succ \xi_i$. We may assume that η does not appear in U. We have $U\phi \Vdash \bot \Leftrightarrow U \star \phi \cdot \pi_0 \in \perp\!\!\!\perp$ (Lemma 5). During the execution of the process $U \star \phi \cdot \pi_0$, the constant η arrives in head position a finite number of times, always through ϕ (since it is deleted each time it arrives in head position), therefore as follows:

$$\eta \star \underline{i} \cdot \mathsf{p} \cdot \vec{\mathsf{q}} \cdot \pi \succ \xi_i \star \pi.$$

Let k be an integer, greater than all the arguments of η during this execution and let $\psi \in \mathbf{\Lambda}$ be such that $\psi \underline{i} \succ \xi_i$ for all $i < k$. Let us set $\tau = \lambda x \lambda \mathsf{p} \lambda \vec{\mathsf{q}}\, \psi x$; thus, we have $\tau \underline{i}\, \mathsf{p} \vec{\mathsf{q}} \succ \psi \underline{i} \succ \xi_i$ for $i < k$. In the process $U \star \phi \cdot \pi_0$, let us replace the constant η by the term τ; we obtain $U \star \psi \cdot \pi_0$. The execution is the same, and therefore $U \star \psi \cdot \pi_0 \in \perp\!\!\!\perp$ and $U\psi \Vdash \bot$. $\blacktriangleleft$

The bar recursion operator

We define below two *proof-like terms* χ and Ψ (which are, in fact, closed λ-terms). In these definitions, the variables i, k represent (intuitively) integers and the variable f represents a function of domain $\mathbb{N}$, with arbitrary values in $\mathbf{\Lambda}$.

- We want a λ-term χ such that:

$$\chi \underline{k} f z \underline{i} \succ f \underline{i} \text{ if } i < k; \chi \underline{k} f z \underline{i} \succ z \text{ if } i \geq k\,.$$

Therefore, we set:

$$\chi = \lambda k \lambda f \lambda z \lambda i((i{<}k)(f)i)z$$

where the boolean $(i{<}k)$ is defined by:

$$(i{<}k) = ((kA)\lambda d\,\mathbf{0})(iA)\lambda d\,\mathbf{1}$$

with $\mathbf{0} = \lambda x \lambda y\, y$ or KI, $\mathbf{1} = \lambda x \lambda y\, x$ or K and $A = \lambda x \lambda y\, yx$ or CI.
The term $\chi \underline{k} f$ is a representation, in λ-calculus, of the finite sequence $(f\underline{0}, f\underline{1}, \ldots, f\underline{k-1})$.
We want a λ-term Ψ such that:

$$\Psi g u \underline{k} f \succ (u)(\chi \underline{k} f)(g)\lambda z(\Psi g u \underline{k}^+)(\chi)\underline{k} f z$$

where $k^+ = ((\mathsf{BW})(\mathsf{C})(\mathsf{B})\mathsf{BB})k$ or $\lambda f \lambda x(f)(k)fx$ is the successor of the integer k.
Thus, we set:

$$\Psi = \lambda g \lambda u(\mathsf{Y})\lambda h \lambda k \lambda f(u)(\chi k f)(g)\lambda z(hk^+)(\chi)kfz,,$$

where Y is the Turing fix point operator:

$$\mathsf{Y} = XX \text{ with } X = \lambda x \lambda f(f)(x)xf = (\mathsf{W})(\mathsf{B})(\mathsf{BW})(\mathsf{C})\mathsf{B}\,.$$

The term Ψ will be called the *bar recursion operator*.

3 Realizing countable choice

The *axiom of countable choice* is the following formula:

$$\text{(CC)} \qquad \forall n \exists x\, F[n,x] \to \exists f \forall n^{\text{int}} F[n, f[n]]$$

where $F[n,x]$ is an arbitrary formula of ZF_ε(see [8]), with parameters and two free variables. The notation $f[n]$ stands for $\mathrm{app}(f,n)$ (the functional app has been defined above).

▶ Remark. This is a strong form of countable choice which shows that, in the realizability model $\mathcal{N}$, every countable sequence has the form $n \mapsto f[n]$ for some f. This will be used in Section 5.

▶ **Theorem 9.** $\lambda g \lambda u(\Psi)gu\,\underline{0}\,\underline{0} \Vdash \mathrm{CC}$.

Proof. The axiom of countable choice is therefore realized in the model of ZF associated with the BBC realizability algebra (in fact, it is sufficient that the realizability algebra satisfies the property formulated in the remark following Theorem 8). We write the axiom of countable choice as follows:

$$\text{(CC)} \qquad \forall n \neg \forall x \neg F[n,x], \forall f \neg \forall n^{\text{int}} F[n, f[n]] \to \bot\,.$$

Let $G, U \in \mathbf{\Lambda}$ be such that $G \Vdash \forall n \neg \forall x \neg F[n,x]$ and $U \Vdash \forall f \neg \forall n^{\text{int}} F[n, f[n]]$. We set $H = \Psi GU$ and we have to show that $H\underline{0}\,\underline{0} \Vdash \bot$. In fact, we shall show that $H\underline{0}\xi \Vdash \bot$ for every $\xi \in \mathbf{\Lambda}$.

▶ **Lemma 10.** *Let $k \in \mathbb{N}$ and $\phi \in \mathbf{\Lambda}$ be such that $(\forall i < k)\exists a_i(\phi \underline{i} \Vdash F[i, a_i])$. If $H\underline{k}\,\phi \nVdash \bot$, then there exist a set a_k and a term $\zeta_{k,\,\phi} \in \mathbf{\Lambda}$ such that:*

$$\zeta_{k,\,\phi} \Vdash F[k, a_k] \quad \textit{and} \quad (H\underline{k}^+)(\chi)\underline{k}\,\phi\,\zeta_{k,\,\phi} \nVdash \bot\,.$$

Proof. Define $\eta_{k,\,\phi} = \lambda z(H\underline{k}^+)(\chi)\underline{k}\,\phi z$, so that $H\underline{k}\,\phi \succ (U)(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi}$.

If $\eta_{k,\,\phi} \Vdash \forall x \neg F[k,x]$ then, by hypothesis on G, we have $G\eta_{k,\,\phi} \Vdash \bot$. Let us check that:

$$(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi} \Vdash \forall n^{\mathrm{int}} F[n, f_k[n]]$$

where f_k is defined by: $f_k[i] = a_i$ if $i < k$ (i.e. $i \in k$); $f_k[i] = \emptyset$ if $i \notin k$.

Indeed, if we set $\phi' = (\chi\underline{k}\phi)(G)\eta_{k,\,\phi}$, we have:

$$\phi'\underline{i} \succ \phi\underline{i} \Vdash F[i,a_i] \text{ for } i < k \text{ and } \phi'\underline{i} \succ (G)\eta_{k,\,\phi} \Vdash \bot$$

for $i \geq k$, and therefore $\phi'\underline{i} \Vdash F[i,\emptyset]$. By hypothesis on U, it follows that $(U)(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi} \Vdash \bot$, in other words $H\underline{k}\,\phi \Vdash \bot$. Thus, we have shown that, if $H\underline{k}\,\phi \nVdash \bot$, then $\eta_{k,\,\phi} \nVdash \forall x \neg F[k,x]$, which gives immediately the desired result. ◀

Let $\phi_0 \in \mathbf{\Lambda}$ be such that $H\underline{0}\phi_0 \nVdash \bot$. By means of Lemma 10, we define $\phi_{k+1} \in \mathbf{\Lambda}$ and a_k recursively on k, by setting $\phi_{k+1} = \chi\underline{k}\,\phi_k\,\zeta_{k,\,\phi_k}$. By definition of χ, we have $\phi_{k+1}\underline{i} \succ \zeta_{k,\phi_k}$ for $i \geq k$. Then, we show easily, by recurrence on k:

$$\phi_{k+1}\underline{i} \succ \phi_{i+1}\underline{i} \succ \zeta_{i,\phi_i} \Vdash F[i,a_i] \text{ for } i \leq k; H\underline{k}\phi_k \nVdash \bot\,.$$

Therefore, we can define:

a function f of domain $\mathbb{N}$ such that $f[i] = a_i$ for every $i \in \mathbb{N}$;

and, by Theorem 8, a term $\phi \in \mathbf{\Lambda}$ such that $\phi\underline{k} \succ \zeta_{k,\,\phi_k}$ for all $k \in \mathbb{N}$. Therefore, we have $\phi\underline{i} \Vdash F[i,f[i]]$ for every $i \in \mathbb{N}$, that is to say $\phi \Vdash \forall n^{\mathrm{int}} F[n,f[n]]$. By hypothesis on U, it follows that $U\phi \Vdash \bot$. Therefore, by Theorem 8, applied to the sequence $\xi_i = \zeta_{i,\,\phi_i}$, there exists an integer k such that $U\psi \Vdash \bot$, for every term $\psi \in \mathbf{\Lambda}$ such that $\psi\underline{i} \succ \zeta_{i,\,\phi_i}$ for $i < k$. hus, in particular, we have $(U)(\chi\underline{k}\,\phi_k)\xi \Vdash \bot$ for every $\xi \in \mathbf{\Lambda}$. Now, by definition of H, we have $H\underline{k}\,\phi_k \succ (U)(\chi\underline{k}\,\phi_k)\xi$ with $\xi = (G)\lambda z(H\underline{k}^+)(\chi)\underline{k}\,\phi_k z$, and therefore $H\underline{k}\,\phi_k \Vdash \bot$, that is a contradiction. Thus, we have shown that $H\underline{0}\phi_0 \Vdash \bot$ for every $\phi_0 \in \mathbf{\Lambda}$. ◀

4 Realizing dependent choice

The *axiom of dependent choice* is the following formula:

$$\text{(DC)} \qquad \forall x \exists y\, F[x,y] \to \exists f \forall n^{\mathrm{int}} F[f[n], f[n+1]]$$

where $F[x,y]$ is an arbitrary formula of ZF_ε, with parameters and two free variables. The notation $f[n]$ stands for $\mathrm{app}(f,n)$ as defined above.

▶ **Theorem 11.** $\lambda g \lambda u (\Psi) g u\,\underline{0}\,\underline{0} \Vdash \mathrm{DC}$.

▶ Remark. The axiom of dependent choice is therefore realized in the model of ZF associated with the BBC realizability algebra (or, more generally, with any realizability algebra satisfying the property formulated in the remark after Theorem 8).

Proof. The proof of Theorem 11 is almost the same as Theorem 9. We write the *axiom of dependent choice* as follows:

$$\text{(DC)} \qquad \forall x\, \neg\forall y\, \neg F[x,y],\ \forall f \neg\forall n^{\mathrm{int}} F[f[n], f[n+1]] \to \bot.$$

Let $G, U \in \mathbf{\Lambda}$ be such that $G \Vdash \forall x\, \neg\forall y\, \neg F[x,y]$ and $U \Vdash \forall f \neg\forall n^{\mathrm{int}} F[f[n], f[n+1]]$. We set $H = \Psi G U$ and we have to show that $H\underline{0}\,\underline{0} \Vdash \bot$. In fact, we shall show that $H\underline{0}\xi \Vdash \bot$ for every $\xi \in \mathbf{\Lambda}$.

▶ **Lemma 12.** *Let $a_0, \ldots, a_k$ be a finite sequence in $\mathcal{M}$ and $\phi \in \mathbf{\Lambda}$ be such that $(\forall i < k)(\phi \underline{i} \Vdash F[a_i, a_{i+1}])$. If $H\underline{k}\,\phi \nVdash \bot$, then there exist $\zeta \in \mathbf{\Lambda}$ and a_{k+1} in $\mathcal{M}$ such that:*

$$\zeta \Vdash F[a_k, a_{k+1}] and (H\underline{k}^+)(\chi)\underline{k}\,\phi\,\zeta \nVdash \bot\,.$$

Proof. Define $\eta_{k,\,\phi} = \lambda z(H\underline{k}^+)(\chi)\underline{k}\,\phi z$, so that $H\underline{k}\,\phi \succ (U)(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi}$. If $\eta_{k,\,\phi} \Vdash \forall y\, \neg F[a_k, y]$ then, by hypothesis on G, we have $G\eta_{k,\,\phi} \Vdash \bot$. We check that:

$$(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi} \Vdash \forall n^{\mathrm{int}} F[f_k[n], f_k[n+1]]$$

where f_k is defined by $f_k[i] = a_i$ for $i \leq k$ (i.e. $i \in k+1$); $f_k[i] = \emptyset$ for $i \notin k+1$. Indeed, if we set $\phi' = (\chi k \phi)(G)\eta_{k,\,\phi}$, we have: $\phi'\underline{i} \succ \phi\underline{i} \Vdash F[a_i, a_{i+1}]$ for $i < k$ and $\phi'\underline{i} \succ (G)\eta_{k,\,\phi} \Vdash \bot$ for $i \geq k$. Therefore, we have $\phi'\underline{i} \Vdash F[f_k[i], f_k[i+1]]$ for every $i \in \mathbb{N}$. By hypothesis on U, it follows that $(U)(\chi\underline{k}\,\phi)(G)\eta_{k,\,\phi} \Vdash \bot$, that is $H\underline{k}\,\phi \Vdash \bot$. Thus, we have shown that, if $H\underline{k}\,\phi \nVdash \bot$, then $\eta_{k,\,\phi} \nVdash \forall y\, \neg F[a_k, y]$, which gives immediately the desired result. ◀

Let $\phi_0 \in \mathbf{\Lambda}$ be such that $H\underline{0}\phi_0 \nVdash \bot$ and let $a_0 = \emptyset$. Using Lemma 12, we define $\phi_{k+1} \in \mathbf{\Lambda}$ and a_{k+1} in $\mathcal{M}$ recursively on k, by setting $\phi_{k+1} = \chi\underline{k}\,\phi_k\,\zeta_{k,\,\phi_k}$, where $\zeta_{k,\,\phi_k}$ is given by Lemma 12, where we set $\phi = \phi_k$. By definition of χ, we have $\phi_{k+1}\underline{i} \succ \zeta_{k,\phi_k}$ for $i \geq k$.

Then, we show easily, by recurrence on k:

$$\phi_{k+1}\underline{i} \succ \phi_{i+1}\underline{i} \succ \zeta_{i,\phi_i} \Vdash F[a_i, a_{i+1}] \text{ for } i \leq k; H\underline{k}\phi_k \nVdash \bot.$$

Therefore, we can define:

$$\text{a function } f \text{ of domain } \mathbb{N} \text{ such that } f[i] = a_i \text{ for every } i \in \mathbb{N}\,;$$

and, by means of Theorem 8, a term $\phi \in \mathbf{\Lambda}$ such that $\phi\underline{k} \succ \zeta_{k,\,\phi_k}$ for every $k \in \mathbb{N}$. Thus, we have $\phi\underline{i} \Vdash F[f[i], f[i+1]]$ for every $i \in \mathbb{N}$, that is to say $\phi \Vdash \forall n^{\mathrm{int}} F[f[n], f[n+1]]$.

By hypothesis on U, it follows that $U\phi \Vdash \bot$. Therefore, by Theorem 8, applied to the sequence $\xi_i = \zeta_{i,\,\phi_i}$, there exists an integer k such that $U\psi \Vdash \bot$, for every term $\psi \in \mathbf{\Lambda}$ such that $\psi\underline{i} \succ \zeta_{i,\,\phi_i}$ for $i < k$. Thus, in particular, we have $(U)(\chi\underline{k}\,\phi_k)\xi \Vdash \bot$ for every $\xi \in \mathbf{\Lambda}$. But, by definition of H, we have $H\underline{k}\,\phi_k \succ (U)(\chi\underline{k}\,\phi_k)\xi$ with $\xi = (G)\lambda z(H\underline{k}^+)(\chi)\underline{k}\,\phi_k z$, and therefore $H\underline{k}\,\phi_k \Vdash \bot$, that is a contradiction.

Thus, we have shown that $H\underline{0}\phi_0 \Vdash \bot$ for every $\phi_0 \in \mathbf{\Lambda}$. ◀

5 Well ordering on R and continuum hypothesis

In this section, we use the notations and the results of [9] and [10]. If F is a closed formula of ZF_ε, the notation $\Vdash F$ means that there exists a proof-like term $\theta \in \mathsf{PL}_0$ (i.e. a closed λ_c-term) such that $\theta \Vdash F$. In Section 3, we have realized the axiom of countable choice (CC). We replace $F[n, x]$ with $\mathrm{int}(n) \to F[n, x]$ and we add a parameter ϕ; we obtain:

$$\Vdash \forall \phi \left(\forall n^{\mathrm{int}} \exists x\, F[n, x, \phi] \to \exists f \forall n^{\mathrm{int}} F[n, f[n], \phi] \right)$$

for every formula $F[n, x, \phi]$ of ZF_ε. In particular, taking $\phi\,\varepsilon\, 2^{\mathbb{N}}$ and $F[n, x, \phi] \equiv (x = \phi(n)) \wedge (x = 0 \vee x = 1)$ (i.e. $(n, x)\,\varepsilon\,\phi \wedge (x = 0 \vee x = 1)$), we find:

$$\Vdash (\forall \phi\,\varepsilon\, 2^{\mathbb{N}}) \exists f \forall n^{\mathrm{int}} \left((f[n] = \phi(n)) \wedge (f[n] = 0 \vee f[n] = 1) \right)\,.$$

For any set f in the ground model $\mathcal{M}$, let $g = \{x\ ;\ f[x] = 1\}$. We have trivially $\mathsf{I} \Vdash \langle n \in g\rangle = \langle f[n] = 1\rangle$. [1] It follows that: $\Vdash \forall f \exists g \forall n\, ((f[n] = 0 \vee f[n] = 1) \to f[n] = \langle n \in g\rangle)$. We have shown that: $\Vdash (\forall \phi\, \varepsilon\, 2^{\mathbb{N}}) \exists g \forall n^{\mathrm{int}} (\phi(n) = \langle n \in g\rangle)$.

Now, in [10], we have built an ultrafilter $\mathcal{D} : \beth 2 \to 2$ on the Boolean algebra $\beth 2$, with the following property: the model $\mathcal{N}$, equipped with the binary relations $\mathcal{D}(\langle x \in y\rangle)$, $\mathcal{D}(\langle x = y\rangle)$, is a model of ZF, denoted $\mathcal{M}_{\mathcal{D}}$, which is an elementary extension of the ground model $\mathcal{M}$. Moreover, $\mathcal{M}_{\mathcal{D}}$ is isomorphic to a transitive submodel of $\mathcal{N}$ (considered as a model of ZF), which contains every ordinal of $\mathcal{N}$. $\mathcal{M}_{\mathcal{D}}$ satisfies the axiom of choice, because we suppose that $\mathcal{M} \models \mathrm{ZFC}$. If we suppose that $\mathcal{M} \models \mathrm{V} = \mathrm{L}$, then $\mathcal{M}_{\mathcal{D}}$ is isomorphic to the class $L^{\mathcal{N}}$ of constructible sets of $\mathcal{N}$. For every $\phi : \mathbb{N} \to 2$, we have obviously $\mathcal{D}(\phi(n)) = \phi(n)$. It follows that:

$$\Vdash (\forall \phi\, \varepsilon\, 2^{\mathbb{N}}) \exists g \forall n^{\mathrm{int}}\, (\phi(n) = \mathcal{D}\langle n \in g\rangle)\,.$$

This shows that the subset of $\mathbb{N}$ defined by ϕ is in the model $\mathcal{M}_{\mathcal{D}}$: indeed, it is the element g of this model. We have just shown that *$\mathcal{N}$ and $\mathcal{M}_{\mathcal{D}}$ have the same reals.*

Therefore, $\mathbb{R}$ is well ordered in $\mathcal{N}$, and we have: $\Vdash$ ($\mathbb{R}$ *is well ordered*). Moreover, if the ground model $\mathcal{M}$ satisfies $\mathrm{V} = \mathrm{L}$, we have: $\Vdash$ (*every real is constructible*). Therefore, the continuum hypothesis is realized.

Since the models $\mathcal{N}$ and $\mathcal{M}_{\mathcal{D}}$ have the same reals, every formula of analysis (closed formula with quantifiers restricted to $\mathbb{N}$ or $\mathbb{R}$) has the same truth value in $\mathcal{M}_{\mathcal{D}}$, $\mathcal{M}$ or $\mathcal{N}$. It follows that:

For every formula F of analysis, we have $\mathcal{M} \models F$ if and only if $\Vdash F$.

In particular, we have $\Vdash F$ or $\Vdash \neg F$.

References [7, 8, 9, 10] are available at `http://www.irif.univ-paris-diderot.fr/~krivine/`.

References

1. S. Berardi, M. Bezem, and T. Coquand. On the computational content of the axiom of choice. *J. Symb. Logic*, 63(2):600–622, 1998.
2. U. Berger and P. Oliva. Modified bar recursion and classical dependent choice. In Springer, editor, *Proc. Logic Colloquium 2001*, pages 89–107, 2005.
3. H.B. Curry and R. Feys. *Combinatory Logic*. North-Holland, 1958.
4. E. Engeler. Algebras and combinators. *Algebra Universalis*, 13(1):389–392, 1981.
5. T. Griffin. A formulæ-as-type notion of control. In *Proc. of the 17th ACM. Symp. on Principles of Progr. Languages*, pages 47–58, 1990. `doi:10.1145/96709.96714`.
6. W. Howard. The formulas–as–types notion of construction. In *Essays on combinatory logic, λ-calculus, and formalism*, pages 479–490. Acad. Press, 1980.
7. J.-L. Krivine. Realizability algebras: a program to well order $\mathbb{R}$. *Logical Methods in Computer Science*, 7(3:02):1–47, 2011.
8. J.-L. Krivine. Realizability algebras II: new models of ZF + DC. *Logical Methods in Computer Science*, 8(1:10):1–28, 2012.
9. J.-L. Krivine. Realizability algebras III: some examples, 2012 (to appear in Math. Struct. Comp. Sc.). URL: `http://arxiv.org/abs/1210.5065`.

[1] The notations $\beth 2$ and $\langle F\rangle$ where F is a closed formula of ZF, with parameters in the realizability model $\mathcal{N}$, are defined in [9, 10]. $\beth 2$ is called the *characteristic Boolean algebra of $\mathcal{N}$*. We have $\langle F\rangle\, \varepsilon\, \beth 2$.

10 J.-L. Krivine. On the structure of classical realizability models of ZF, 2014 (to appear in Proceedings Types 2014). URL: `http://arxiv.org/abs/1408.1868`.

11 C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In *Recursive function theory: Proc. Symp. in pure math. vol. 5, Amer. Math. Soc. Providence, Rhode Island*, pages 1–27, 1962.

12 T. Streicher. A classical realizability model arising from a stable model of untyped λ-calculus, 2013 (to appear).

Extracting Non-Deterministic Concurrent Programs

Ulrich Berger

Swansea University, U.K.

Abstract

We introduce an extension of intuitionistic fixed point logic by a modal operator facilitating the extraction of non-deterministic concurrent programs from proofs. We apply this extension to program extraction in computable analysis, more precisely, to computing with Tsuiki's infinite Gray code for real numbers.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages, F.4.1 Mathematical Logic

Keywords and phrases Proof theory, realizability, program extraction, non-determinism, concurrency, computable analysis

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.26

1 Introduction

The fact that proofs in constructive systems carry computational content is known as the Brouwer-Heyting-Kolmogorov interpretation or the Curry-Howard correspondence. It is the origin of various methods to automatically extract certified programs from formal proofs which are implemented in proof systems such as Nuprl [8], PX [9], Coq [12], Isabelle [5], Agda [7], Minlog [3]. The extracted programs are usually functional; other programming paradigms, such as non-determinism or concurrency, are hardly covered by this methodology. This may be considered a weakness of program *extraction* compared with existing program *verification* techniques which *do* cover these programming paradigms. This paper aims to narrow the gap between program extraction and verification by introducing *Concurrent Fixed Point Logic* (CFP), an intuitionistic theory of inductive and coinductive definitions extended by a modal operator enabling the extraction of non-deterministic concurrent programs.

The development of CFP was triggered by an example from computable analysis, *Tsuiki's infinite Gray code* for real numbers [14], which encodes a real number $x \in \mathbb{I} = [-1, 1]$ [1] by the itinerary of x along the *tent map*

$$\mathrm{t} : \mathbb{I} \to \mathbb{I}, \quad \mathrm{t}(x) = 1 - 2|x|.$$

More precisely, x is encoded by the stream $a_0 : a_1 : a_2 : \ldots$ where the head of the stream, a_0, equals 0, 1 or $\perp$ (= undefined) depending on whether x is less, greater, or equal to 0, and the tail of the stream, $a_1 : a_2 : \ldots$, encodes $\mathrm{t}(x)$. Since $\mathrm{t}(0) = 1$ and $\mathrm{t}(1) = \mathrm{t}(-1) = -1$, at most one a_i can be undefined, and in that case $a_{i+1} = 1$ and $a_k = 0$ for all $k > i + 1$.

Infinite Gray code stands out from other encodings of real numbers by the fact that it is at the same time *unique* (every real number in $\mathbb{I}$ has exactly one Gray code) and *computable*

[1] Tsuiki considers reals in the interval $[0, 1]$, but we find it convenient to work with an interval that is symmetric around 0.

© Ulrich Berger;
licensed under Creative Commons License CC-BY

(it is computably equivalent to admissible encodings such as the Cauchy- or the signed digit representation). In contrast, other computable representations of the reals are highly redundant. For example, a signed digit representation of $x \in \mathbb{I}$ is any stream $d_0 : d_1 : d_2 : \ldots$ of signed digits $d_i \in \mathrm{SD} = \{-1, 0, 1\}$ such that $x \in \mathbb{I}_{d_0} := [d_0/2 - 1/2, d_0/2 + 1/2]$ and $d_1 : d_2 : \ldots$ is a signed digit representation of $2x - d_0$. It is easy to see that every non-dyadic real in $\mathbb{I}$ has continuum many signed digit representations. The infinite Gray code's unlikely combination of uniqueness and computability is, of course, only possible because it is not total: dyadic rationals in $]-1, 1[$ have an infinite Gray code with one undefined digit.

Tsuiki defines the following function transforming infinite Gray code into signed digit representation (`a0:a1:...:an:s` stands for a stream that begins with the digits `a0:a1:...:an` and continues with the stream `s`; we also use the function `swap 0 = 1; swap 1 = 0`):

```
gtos (0:s)     = -1 : gtos s
gtos (1:b:s)   =  1 : gtos (swap b : s)
gtos (a:1:c:s) =  0 : gtos (a : swap c : s)
```

Since the pattern `a:1:c:s` in the third line overlaps with those in the first and second line, this definition cannot be executed in a deterministic functional language, but should rather be viewed as a system of rewrite rules from left to right. Tsuiki introduced (and implemented in Prolog) two-head Turing machines that are able to execute definitions as the one above: Initially, the machine's first head reads the first input digit and its second head reads the second input digit. If the computation of the first input digit terminates first, the first or second line fires, if the computation of the second input digit terminates first, the third line fires and the second head moves one position to the right (while the first head continues to wait for the computation of the first input digit to terminate, which may or may not happen eventually). This is an example of a non-deterministic concurrent computation with potentially incompatible results of the different threads. One can show that no continuous function can translate Gray code into signed digit representation. The closest to infinite Gray code one seems to get with traditional program extraction is a program that works on so-called pre Gray code, i.e., streams representing constructions of infinite Gray code [4].

In this paper, we extend the method of program extraction developed in [1] in order to be able to extract concurrent programs such as `gtos` above from proofs of their specification. Our basic formal system is essentially the one considered in [1] and is called IFP in this paper (Intuitionistic Fixed Point Logic). It comprises intuitionistic first-order logic with inductive and coinductive definitions and a realizability interpretation (Sect. 2). An important feature of program extraction in IFP is the fact that proofs can be carried out in any non-computational theory (Sect. 2). The correctness of the extracted programs will then be proven in the same theory. This makes it possible to extract programs from proofs in an abstract axiomatic setting.

Our leading example is the structure $\mathbb{R}$ of the real numbers with 0, 1, addition and multiplication, which we specify in IFP by the disjunction-free axioms of Archimedean ordered fields (replacing $\vee$ by $\neg \wedge \neg$ if required). The set of natural numbers $\mathbb{N}$ can be defined in IFP as the least subset of the reals that contains 0 and is closed under the $+1$ function. Hence IFP includes Heyting Arithmetic. We define predicates C and G as the largest predicates on $\mathbb{I}$ satisfying

$$\begin{aligned} \mathrm{C}(x) &\leftrightarrow \bigvee_{d \in \mathrm{SD}} x \in \mathbb{I}_d \wedge \mathrm{C}(2x - d) \\ \mathrm{G}(x) &\leftrightarrow (x \neq 0 \to x < 0 \vee x > 0) \wedge \mathrm{G}(\mathrm{t}(x)) \end{aligned}$$

The signed digit representations of x will be the realizers of $\mathrm{C}(x)$, and the Gray code of x will be a realizer of $\mathrm{G}(x)$. Hence, in order to extract a program that translates Gray code into signed digit representation, we may attempt to prove $\mathrm{G} \subseteq \mathrm{C}$. Since there can be no deterministic program accomplishing the transformation, this proof cannot be carried out in IFP alone, but some extra principle is needed. The following *Disjunction Principle*

$$\text{(DP)} \quad (A \overset{P}{\vee} B) \wedge (P \overset{Q}{\vee} C) \to (A \vee B \vee C)$$

suffices, where $A \overset{P}{\vee} B$ is shorthand for $(P \to A \vee B) \wedge (\neg P \to A \wedge B)$ and P, Q, A, B, C range over non-computational formulas. Using (DP) (with $P := x \neq 0$, $Q := x \neq 0$, $A := x \in \mathbb{I}_{-1}$, $B := x \in \mathbb{I}_1$, and $C := x \in \mathbb{I}_0$), we show $\mathrm{G} \subseteq \mathrm{C}$ (Theorem 12). In order to extract a program from the proof of $\mathrm{G} \subseteq C$ one needs a realizer of (DP) which, unfortunately, does not exist in IFP. To solve this problem, we embed, in Sect. 5, IFP into Concurrent Fixed Point Logic (CFP), where (DP) *is* realizable (Lemma 30). CFP extends the language of IFP by a modal operator S and extends programs by families $\mathrm{Fam}\,\varphi$ of programs $\varphi(i)$ (where i ranges over a countable index set $\mathcal{I}$) that are to be executed non-deterministically and concurrently. This is expressed by an operational semantics (Sect. 6) that allows $\mathrm{Fam}\,\varphi$ to reduce to $\varphi(i)$ for every $i \in \mathcal{I}$. A family $\mathrm{Fam}\,\varphi$ realizes a formula of the form $\mathrm{S}(A)$, where A is computational, if (i) $\varphi(i)$ yields a result for at least one $i \in \mathcal{I}$ and (ii) for any $i \in \mathcal{I}$, if $\varphi(i)$ yields a result b, then b realizes A (Sect. 5). The proof rules for CFP given in Sect. 5 endow the modality S with the structure of a strong monad which can be seen as a proof-theoretic analogue of Moggi's monadic computational lambda-calculus [10]. With some coding effort the countably infinite non-determinism represented by the construct $\mathrm{Fam}\,\varphi$ can be simulated by binary non-deterministic choice as considered, for example, in [6]. The denotational semantics introduced in Sect 6 could be simplified accordingly. In order not to distract the readers attention from the main issues of this paper, we refrain from carrying out these simplifications.

There exists a rich literature on modelling and verifying non-deterministic and concurrent programs (e.g. [13, 11, 6] and many others). The novelty of our work lies in the fact that we extract these programs from ordinary mathematical proofs together with a formal certificate of their correctness.

2 Intuitionistic Fixed Point Logic

We briefly recall (and slightly improve) the system IFP of Intuitionistic Fixed Point Logic and its realizability interpretation as defined in [1].

IFP is intuitionistic first-order predicate logic with inductive and coinductive definitions given as least and greatest fixed points of strictly positive predicate transformers. The *formulas* of IFP, are $P(\vec{t})$, $X(\vec{t})$, $A \wedge B$, $A \vee B$, $A \to B$, $\forall x\, A$, $\exists x\, A$, $(\mu\,\Phi)(\vec{t})$ (inductive definitions) and $(\nu\,\Phi)(\vec{t})$ (coinductive definitions), where P ranges over predicate constants, X ranges over predicate variables, $\vec{t}$ ranges over tuples of first-order terms of a given signature, x ranges over object variables and Φ ranges over strictly positive predicate transformers. The latter are of the form $\lambda X \lambda \vec{x}\,.\, A$ where $\vec{x}$ and $\vec{t}$ have the same lengths and A is strictly positive in X i.e. X does not occur free in any premise of a subformula of A which is an implication [2]. Predicate constants and variables have fixed arities. We assume that there is a 0-ary predicate constant $\bot$ for falsity and an equality predicate.

A *predicate* is either a predicate constant P, a predicate variable X, an abstraction $\lambda \vec{x}.A$, an inductive predicate $\mu\,\Phi$, or a coinductive predicate $\nu\,\Phi$. The application, $\mathcal{P}(\vec{t})$, of

[2] In [2] it is shown that this condition can be weakened to provable monotonicity.

a predicate $\mathcal{P}$ to a list of terms $\vec{t}$ is a primitive syntactic construct, except when $\mathcal{P}$ is a an abstraction, $\lambda\vec{x}\,A$, in which case $\mathcal{P}(\vec{t})$ stands for $A[\vec{t}/\vec{x}]$. We will use abbreviations such as $\mathcal{P} \subseteq \mathcal{Q}$ for $\forall\vec{x}\,.\,\mathcal{P}(\vec{x}) \to \mathcal{Q}(\vec{x})$ and will sometimes write $\{x \mid A\}$ for $\lambda x\,A$, etc. We will also write $\mathcal{P}(\vec{x}) \stackrel{\mu}{=} A[\mathcal{P}/X]$ for $\mathcal{P} = \mu\,\lambda X \lambda\vec{x}\,A$ and similarly for ν.

The *proof rules* of IFP are the usual ones of intuitionistic predicate calculus with equality augmented by rules expressing that $\mu\,\Phi$ and $\nu\,\Phi$ are the least and greatest fixed points of the operator Φ (as is well-known, the fixed point property can be replaced by inclusions):

$$\frac{}{\Gamma \vdash \Phi(\mu\,\Phi) \subseteq \mu\,\Phi}\ \text{Closure} \qquad \frac{\Gamma \vdash \Phi(\mathcal{P}) \subseteq \mathcal{P}}{\Gamma \vdash \mu\,\Phi \subseteq \mathcal{P}}\ \text{Induction}$$

$$\frac{}{\Gamma \vdash \nu\,\Phi \subseteq \Phi(\nu\,\Phi)}\ \text{Coclosure} \qquad \frac{\Gamma \vdash \mathcal{P} \subseteq \Phi(\mathcal{P})}{\Gamma \vdash \mathcal{P} \subseteq \nu\,\Phi}\ \text{Coinduction}$$

Realizability is formalized in an extension RIFP of IFP by an extra sort of realizers and *program terms* (*programs for short*) of this new sort. Programs are untyped λ-terms with pairing, injections and recursion, more precisely, variables $a, b, c, d, e, f, g, \ldots$, the constant nil, and the composite terms $\langle M, N\rangle$, $\mathrm{inl}(M)$, $\mathrm{inr}(M)$, $\lambda a.M$, $\pi_i(M)$ $(i = 1, 2)$, case M of $\{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\}$, $(M\,N)$, $\mathrm{rec}\,a\,.\,M$. The free variables of a program are defined as usual (the constructs λa, $\mathrm{rec}\,a$ and $\mathrm{inl}(a) \to$, $\mathrm{inr}(a) \to$ in a case term bind the variable a). In order to keep programs readable, we will use pattern matching in a slightly liberal way by allowing wildcards, nested patterns and possibly omitting patterns in which case realizers matching an omitted pattern are mapped to the default value nil. For example, case M of $\{\mathrm{inl}(\mathrm{inr}(_)) \to N\}$ stands for the nested case analysis case M of $\{\mathrm{inl}(a) \to$ case a of $\{\mathrm{inl}(a_0) \to \mathrm{nil}\,;\,\mathrm{inr}(a_1) \to N\}\,;\,\mathrm{inr}(b) \to \mathrm{nil}\}$ where a and a_1 are not free in N. Closed programs built from nil by pairing $\langle\cdot,\cdot\rangle$ and the injections $\mathrm{inl}(\cdot)$, $\mathrm{inr}(\cdot)$ are called *data*. All axioms and rules for IFP, including closure, induction, coclosure and coinduction and the rules for equality, are extended to RIFP. In addition, we add the equations

$$\begin{aligned}
\pi_i(\langle M_1, M_2\rangle) &= M_i \qquad (i = 1, 2)\\
\text{case } \mathrm{inl}(M) \text{ of } \{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\} &= L[M/a]\\
\text{case } \mathrm{inr}(M) \text{ of } \{\mathrm{inl}(a) \to L\,;\,\mathrm{inr}(b) \to R\} &= R[M/b]\\
(\lambda a.M)N &= M[N/a]\\
\mathrm{rec}\,a\,.\,M &= M[\mathrm{rec}\,a\,.\,M/a]
\end{aligned}$$

We define simultaneously *non-computational* and *faithful* formulas (abbreviated *nc* and *ff*). nc formulas without free predicate variables will be called *ncc* formulas.

- The class of nc formulas contains all atomic formulas (i.e. $P(\vec{t})$ and $X(\vec{t})$) and is closed under all logical operators except disjunction (but including inductive and coinductive definitions). In addition, if A is ff and B is nc, then $A \to B$ is nc.
- The class of ff formulas contains all ncc formulas and is closed under all logical operators except implication, universal quantification and coinductive definitions.

Note that all disjunction-free formulas without free predicate variables (which is the class of formulas called non-computational in [1]) are ncc.

Realizability assigns to every IFP-formula A a unary RIFP-predicate $\mathbf{r}(A)$. Intuitively, the RIFP-formula $\mathbf{r}(A)(a)$, which we will usually write $a\,\mathbf{r}\,A$, states that a "realizes" A. The definition of $a\,\mathbf{r}\,A$ is relative to a fixed one-to-one mapping from IFP-predicate variables X to RIFP-predicate variables $\tilde{X}$ with one extra argument place of the new sort. If the formula A has the free predicate variables $X_1, \ldots, X_n$, then the formula $a\,\mathbf{r}\,A$ has the free predicate

variables $\tilde{X}_1, \ldots, \tilde{X}_n$. For $\Phi = \lambda X \lambda \vec{x}\, A$ we set $\mathbf{r}(\Phi) = \lambda \tilde{X} \lambda (b, \vec{x})\, b\, \mathbf{r}\, A$.

$$
\begin{array}{rrcl}
 & a\,\mathbf{r}\,X(\vec{t}) & = & \tilde{X}(a,\vec{t}) \\
 & a\,\mathbf{r}\,P(\vec{t}) & = & P(\vec{t}) \\
a\,\mathbf{r}\,(A \wedge B) = a\,\mathbf{r}\,(B \wedge A) & & = & (a\,\mathbf{r}\,A) \wedge B \quad \text{if } B \text{ is ncc} \\
 & b\,\mathbf{r}\,(A \to B) & = & A \to b\,\mathbf{r}\,B \quad \text{if } A \text{ is ncc} \\
\text{Otherwise} & c\,\mathbf{r}\,(A \wedge B) & = & \pi_1(c)\,\mathbf{r}\,A \wedge \pi_2(c)\,\mathbf{r}\,B \\
 & c\,\mathbf{r}\,(A \vee B) & = & \exists a\,(c = \mathrm{inl}(a) \wedge a\,\mathbf{r}\,A) \vee \exists b\,(c = \mathrm{inr}(b) \wedge b\,\mathbf{r}\,B) \\
 & f\,\mathbf{r}\,(A \to B) & = & \forall a\,(a\,\mathbf{r}\,A \to (f\,a)\,\mathbf{r}\,B) \\
 & a\,\mathbf{r}\,\diamond x\,A & = & \diamond x\,(a\,\mathbf{r}\,A) \quad (\diamond = \forall, \exists) \\
 & a\,\mathbf{r}\,(\diamond\,\Phi)(\vec{t}) & = & (\diamond\,\mathbf{r}(\Phi))(a,\vec{t}) \quad (\diamond = \mu, \nu)
\end{array}
$$

▶ **Lemma 1.** *If A is an ncc formula, then $a\,\mathbf{r}\,A$ is equivalent to A, provably in* RIFP.

Proof. See Appendix. ◀

▶ **Theorem 2** (Soundness)**.** *From a proof in* IFP *of $\Delta, \Gamma \vdash A$, where Δ consists of ncc formulas, one can extract a program term M such that* RIFP *proves $\Delta, \vec{a}\,\mathbf{r}\,\Gamma \vdash (M\,\vec{a})\,\mathbf{r}\,A$.*

Proof. The proof is as in [1]. ◀

In the following we will apply the Soundness Theorem with Δ being the axioms of Archimedean ordered fields, written as ncc formulas, and Γ a set of formulas for which we have (or construct) programs realizing them. We are free to add stability, $\neg\neg A \to A$, for ncc formulas A to Δ, since, clearly, $\neg\neg A \to A$ is ncc if A is.

3 Cauchy and signed digit representation of real numbers

In this section we introduce real numbers in IFP. We define predicates A and C which correspond, via realizability, to the (fast) Cauchy representation and the signed digit representations, respectively, and prove their equivalence.

We let $(\mathbb{R}, 0, 1, +, \cdot, \leq)$ be the structure of real numbers, of which we will only use that it is an Archimedean ordered field. In order for the axioms to be ncc we write the Dichotomy axiom as $\neg x < y \wedge \neg y < x \to x = y$. As explained at the end of the previous section, we may assume stability of ncc and hence atomic formulas, that is, $\neg\neg x < y \to x < y$, etc. The Archimedean principle, written as an ncc formula, is

$$\text{(AP)} \qquad \forall x\,.\,(\forall n \in \mathbb{N}\,|x| \leq 2^{-n}) \to x = 0$$

where the set of natural numbers, $\mathbb{N} \subseteq \mathbb{R}$, is defined inductively as

$$\mathbb{N}(x) \stackrel{\mu}{=} x = 0 \vee \mathbb{N}(x-1)$$

(recall that this stands for $\mathbb{N} = \mu\,\lambda X \lambda x\,.\,x = 0 \vee X(x-1)$). The integers and the rational numbers, $\mathbb{Z}, \mathbb{Q} \subseteq \mathbb{R}$ are defined by

$$\mathbb{Z} = \{x \mid x \in \mathbb{N} \vee -x \in \mathbb{N}\}, \qquad \mathbb{Q} = \{q \mid \exists k \in \mathbb{N}\,.\,k > 0 \wedge kq \in \mathbb{Z}\}$$

The function $\max : \mathbb{R} \to \mathbb{R}$ is introduced (i.e. added to the signature) and axiomatized by

$$\max(x,y) \leq z \leftrightarrow x \leq z \wedge y \leq z$$

and the absolute value is defined by $|x| := \max(x, -x)$.

The set of real numbers in $\mathbb{I}$ that can be approximated by rational numbers is defined by

$$\mathrm{A} = \{x \in \mathbb{I} \mid \forall n \in \mathbb{N}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\,.\, |x - q| \leq 2^{-n}\}\,.$$

A realizer of "$x \in \mathrm{A}$" is a fast rational Cauchy-sequence converging to x.

Let $\mathrm{SD} = \{-1, 0, 1\}$ be the set of *signed digits* (hence $d \in \mathrm{SD}$ means $d = -1 \vee d = 0 \vee d = 1$) and set $\mathbb{I}_d := [d/2 - 1/2, d/2 + 1/2] \subseteq \mathbb{I}$. We define $\mathrm{C} \subseteq \mathbb{R}$ coinductively by

$$\mathrm{C} \overset{\nu}{=} \{x \in \mathbb{R} \mid \exists d \in \mathrm{SD}\,.\, x \in \mathbb{I}_d \wedge 2x - d \in \mathrm{C}\}\,.$$

Classically, $\mathrm{C} = \mathbb{I}$. A realizer of "$x \in \mathrm{C}$" is a stream of signed digits $d_0, d_1, \ldots$ such that

$$x = \sum_{i=0}^{\infty} d_i 2^{-(i+1)}\,.$$

Therefore, C corresponds to the signed digit representation of real numbers in $\mathbb{I}$.

▶ **Lemma 3.**
(a) *If $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ for some $a, b \in \mathbb{Q}$, then $y \in \mathrm{A}$.*
(b) *If $x \in \mathrm{C}$ and $b \in \mathbb{Q}$ is such that $x + b \in \mathbb{I}$, then $x + b \in \mathrm{C}$.*
(c) *If $x \in \mathrm{C}$ and $2x + b \in \mathbb{I}$, where $b \in \mathbb{Q}$, then $2x + b \in \mathrm{C}$.*
(d) *If $x \in \mathrm{C}$, then $-x \in \mathrm{C}$.*

Proof. See Appendix. ◀

▶ **Theorem 4.** $\mathrm{C} = \mathrm{A}$.

Proof. "$\mathrm{C} \subseteq A$": We show $\forall n \in \mathbb{N} \forall x \in \mathrm{C}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\,.\, |x - q| \leq 2^{-n}$ by induction on n. If $n = 0$, set $q = 0$. For $n+1$ assume $x \in \mathrm{C}$. Let $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in \mathrm{C}$. By i.h. we have $q \in \mathbb{Q} \cap \mathbb{I}$ with $|(2x - d) - q| \leq 2^{-n}$. Hence $|x - (q + d)/2| = |(2x - d) - q|/2 \leq 2^{-(n+1)}$.

For "$A \subseteq \mathrm{C}$" we use coinduction. Assume $x \in A$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in A$. Since $x \in \mathrm{A}$ we have $q \in \mathbb{Q}$ such that $|x - q| \leq 1/4$. If $q \leq -1/2$ we know $x \in \mathbb{I}_{-1}$, and from Lemma 3 (a) it follows that $2x + 1 \in A$. Similarly, if $q \geq 1/2$ we know $x \in \mathbb{I}_1$ and $2x - 1 \in A$. Otherwise $x \in \mathbb{I}_0$ and $2x \in A$. ◀

4 Infinite Gray code

In this section we introduce infinite Gray code via a coinductive predicate G and prove its equivalence to the Cauchy and signed digit representation, using the (in IFP not realizable) Disjunction Principle to prove $\mathrm{G} \subseteq \mathrm{C}$.

Let $\mathrm{t}(x) = 1 - 2|x|$ be the *tent map*, which maps $\mathbb{I}$ onto itself, and set

$$\mathrm{D}(x) := x \neq 0 \to x \leq 0 \vee x \geq 0.$$

We define $\mathrm{G} \subseteq \mathbb{R}$ coinductively by

$$\mathrm{G}(x) \overset{\nu}{=} |x| \leq 1 \wedge \mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))\,.$$

Classically, we clearly have $\mathrm{G} = \mathbb{I}$. A realizer of "$x \in \mathrm{G}$" is a stream of partial Booleans representing the itinerary of x along the tent map. Such a realizing stream can have at most one undefined item, namely if $\mathrm{t}^n(x) = 0$, in which case the item with index n is undefined. In this case, $\mathrm{t}^{(n+1)}(x) = 1$ and $\mathrm{t}^m(x) = 0$ for $m > n + 1$, hence the items after the undefined

one are $1, 0, 0, 0, \ldots$. In the proof of Theorem 5 below we use the axiom of countable choice for rational numbers (AC^ω) and Markov's principle (MP), which are both realizable.

$$(\mathrm{AC}^\omega) \qquad (\forall n \in \mathbb{N}\, \exists q \in \mathbb{Q}\, A(n,q)) \to \exists f : \mathbb{N} \to \mathbb{Q}\, \forall n \in \mathbb{N}\, A(n, f(n)) \text{ where } A \text{ is ncc.}$$

$$(\mathrm{MP}) \qquad (\forall n \in \mathbb{N}\,.\, A(n) \vee \neg A(n)) \wedge (\neg\neg \exists n \in \mathbb{N}\, A(n)) \to \exists n \in \mathbb{N}\, A(n) \text{ where } A \text{ is ncc.}$$

It is easy to see that (AC^ω) is realized by the identity function, $\lambda a\,.\,a$, and (MP) is realized by unbounded search through the (unary representations of) natural numbers, which can be coded as $\lambda f\,.\,(\mathrm{rec}\, g\,.\,\lambda a\,.\,\mathrm{case}\ f\, a\ \mathrm{of}\ \{\mathrm{inl}(_) \to a\,;\, \mathrm{inr}(_) \to g\,(\mathrm{inr}(a))\})\,(\mathrm{inl}(\mathrm{nil}))$.

▶ **Theorem 5** ($\mathrm{AP}, \mathrm{AC}^\omega, \mathrm{MP}$)**.** $\mathrm{A} \subseteq \mathrm{G}$

Proof. By coinduction. Assume $\mathrm{A}(x)$. We have to show $\mathrm{D}(x)$ and $A(\mathrm{t}(x))$. The latter is easy: To show $A(\mathrm{t}(x))$, fix $n \in \mathbb{N}$. Let $q \in \mathbb{Q} \cap \mathbb{I}$ such that $|x - q| \le 2^{-(n+1)}$. Then $\mathrm{t}(q) \in \mathbb{Q} \cap \mathbb{I}$ and $|\mathrm{t}(x) - \mathrm{t}(q)| = 2||x| - |q|| \le 2|x - q| \le 2^{-n}$.

Now we show $\mathrm{D}(x)$. Assume $x \neq 0$. By (AC^ω), there exists a sequence of rational numbers q_n such that $\forall n \in \mathbb{N}\, |x - q_n| \le 2^{-n}$. We show $\neg\forall n \in \mathbb{N}\, |q_n| \le 2^{1-n}$. Assume $\forall n \in \mathbb{N}\, |q_n| \le 2^{1-n}$. Then $\forall n \in \mathbb{N}\, |x| \le 2|q_n| \le 2^{2-n}$. By (AP) it follows $x = 0$, contradicting the assumption $x \neq 0$. By (MP) it follows that there exists $n \in \mathbb{N}$ such that $|q_n| > 2^{1-n}$. If $q_n > 0$, then we have $q_n \ge 2^{1-n}$, which, together with $|x - q_n| \le 2^{-n}$, implies $x \ge 2^{-n} > 0$. Similarly, if $q_n < 0$, then we have $q_n \le -2^{1-n}$, which, together with $|x - q_n| \le 2^{-n}$, implies $x \le -2^{-n} < 0$. ◀

In Lemma 6 below we use the Disjunction Principle discussed in the Introduction:

$$(\mathrm{DP}) \qquad (A \overset{P}{\vee} B) \wedge (P \overset{Q}{\vee} C) \to A \vee B \vee C\,.$$

▶ **Lemma 6** (DP)**.** *If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.*

Proof. Assume $x \in \mathrm{G}$. Then $\mathrm{D}(x)$ and $\mathrm{D}(\mathrm{t}(x))$. In order to apply (DP), set $A := x \in \mathbb{I}_{-1}$, $B := x \in \mathbb{I}_1$, $C := x \in \mathbb{I}_0$, $P := x \neq 0$, and $Q := \mathrm{t}(x) \neq 0$. Hence, it suffices to show that the premises of (DP) hold. $P \to A \vee B$ is $\mathrm{D}(x)$. To show $\neg P \to A \wedge B$, assume $\neg(x \neq 0)$, that is, $x = 0$. Then clearly $x \in \mathbb{I}_{-1}$ and $x \in \mathbb{I}_1$. To show $Q \to C \vee P$, assume $\mathrm{t}(x) \neq 0$. Then $\mathrm{t}(x) \le 0 \vee \mathrm{t}(x) \ge 0$, since $\mathrm{D}(\mathrm{t}(x))$. If $\mathrm{t}(x) \le 0$, then $|x| \ge \frac{1}{2}$, hence $x \neq 0$. If $\mathrm{t}(x) \ge 0$, then $x \in \mathbb{I}_0$. Finally, to show $\neg Q \to C \wedge P$, assume $\neg(\mathrm{t}(x) \neq 0)$, that is, $\mathrm{t}(x) = 0$. Then $|x| = \frac{1}{2}$, hence $x \in \mathbb{I}_0$ and $x \neq 0$. ◀

▶ **Lemma 7.** *If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$ and $|x| \in \mathrm{G}$.*

Proof. Assume $x \in \mathrm{G}$. Then $\mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))$. Since $\mathrm{t}(x) = \mathrm{t}(-x) = \mathrm{t}(|x|)$ it easily follows $\mathrm{D}(-x) \wedge \mathrm{G}(\mathrm{t}(-x))$ and also $\mathrm{D}(-x) \wedge \mathrm{G}(\mathrm{t}(|x|))$. Hence $\mathrm{G}(-x)$ and $\mathrm{G}(|x|)$. ◀

▶ **Lemma 8.** *If $|x| \le 1$ and $\mathrm{G}(\frac{x+1}{2})$, then $\mathrm{G}(x)$. Equivalently, if $0 \le x \le 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x - 1)$.*

Proof. Assume $|x| \le 1$ and $\mathrm{G}(\frac{x+1}{2})$. Then $\mathrm{G}(\mathrm{t}(\frac{x+1}{2}))$, that is, $\mathrm{G}(-x)$, since the assumption $|x| \le 1$ implies $\mathrm{t}(\frac{x+1}{2}) = -x$. Hence $G(x)$, by Lemma 7. ◀

▶ **Lemma 9.** *If $\mathrm{G}(x)$, then $\mathrm{G}(\mathrm{t}(x))$.*

Proof. Obvious. ◀

▶ **Lemma 10.** *If $0 \le x \le 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1 - x)$.*

Proof. Assume $0 \leq x \leq 1$ and $\mathrm{G}(x)$. Since $0 \leq 1 - x \leq 1$, it suffices to show $\mathrm{G}(\mathrm{t}(1-x))$. The assumption $x \leq 1$ implies that $\mathrm{t}(1-x) = 2x - 1$. Hence $\mathrm{G}(2x-1)$, by Lemma 8. ◀

▶ **Lemma 11.** *If* $|x| \leq 1$ *and* $\mathrm{G}(\frac{x}{2})$*, then* $\mathrm{G}(x)$*. Equivalently, if* $-\frac{1}{2} \leq x \leq \frac{1}{2}$ *and* $\mathrm{G}(x)$*, then* $\mathrm{G}(2x)$*.*

Proof. Assume $|x| \leq 1$ and $\mathrm{G}(\frac{x}{2})$. Hence $\mathrm{G}(\mathrm{t}(\frac{x}{2}))$. Since $\mathrm{t}(\frac{x}{2}) = 1 - |x|$, we have $\mathrm{G}(1-|x|)$ and therefore $\mathrm{G}(|x|)$, by Lemma 10 and since $0 \leq 1 - |x| \leq 1$. Hence $\mathrm{G}(\mathrm{t}(x))$ (since $\mathrm{t}(|x|) = \mathrm{t}(x)$), by Lemma 9. Furthermore $\mathrm{D}(\frac{x}{2})$ which implies $\mathrm{D}(x)$. It follows $\mathrm{G}(x)$. ◀

▶ **Theorem 12** (DP)**.** $\mathrm{G} \subseteq \mathrm{C}$.

Proof. By coinduction. Assume $\mathrm{G}(x)$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $\mathrm{G}(2x-d)$. By Lemma 6 (which was proven using DP), there exists $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$. By the Lemmas 9 (note that if $x \in \mathbb{I}_{-1}$, then $2x+1 = \mathrm{t}(x)$), 8, and 11 we have $\mathrm{G}(2x-d)$. ◀

5 Concurrent Fixed Point Logic

In this section we introduce Concurrent Fixed Point Logic, CFP, extending IFP.

CFP extends the language of IFP by a modal operator S. The proof calculus of IFP is extended by the rules

$$\frac{\Gamma \vdash A}{\Gamma \vdash \mathrm{S}(A)}\ (\mathrm{S}^+) \qquad \frac{\Gamma \vdash \mathrm{S}(A) \qquad \Gamma, A \vdash \mathrm{S}(B)}{\Gamma \vdash \mathrm{S}(B)}\ (\mathrm{S}^-) \qquad \frac{\Gamma \vdash \mathrm{S}(A)}{\Gamma \vdash A}\ (\mathrm{S}^{\mathrm{nc}}) \text{ if } A \text{ is ncc}$$

which will be justified by the Soundness Theorem 16. The rules (S^+) and (S^-) state that S is a strong monad (see [10] for an analogous construction for a computational lambda-calculus). They immediately imply monotonicity, $(A \to B) \to \mathrm{S}(A) \to \mathrm{S}(B)$, and idempotency, $\mathrm{S}(\mathrm{S}(A)) \leftrightarrow \mathrm{S}(A)$, and that S interacts nicely with the logical operators, for example, $\mathrm{S}(A \wedge B) \leftrightarrow \mathrm{S}(A) \wedge \mathrm{S}(B)$, $\mathrm{S}(A \to B) \to \mathrm{S}(A) \to \mathrm{S}(B)$ and $\mathrm{S}(\forall x\, A) \to \forall x\, \mathrm{S}(A)$.

The modality S can be viewed as a predicate transformer by setting $\mathrm{S}(\mathcal{P}) = \lambda \vec{x}\,.\, S(\mathcal{P}(\vec{x}))$. In particular, $\mathrm{S}(\lambda \vec{x}\,.\, A) = \lambda \vec{x}\,.\, S(A)$. Hence, if $\Phi = \lambda X \lambda \vec{x}\,.\, A$ is a monotone predicate transformer, we can form the composition $\Phi \circ S = \lambda X\,.\, \Phi(\mathrm{S}(X)) = \lambda X \lambda \vec{x}\,.\, A[\mathrm{S}(X)/X] = \lambda X \lambda \vec{x}\,.\, A[\lambda \vec{y}\,.\, \mathrm{S}(X(\vec{y}))/X]$. We call a predicate variable *guarded* in a formula A if every free occurrence of X in A is within a disjunctive or existential subformula of A. We call a predicate transformer $\Phi = \lambda X \lambda \vec{x}\,.\, A$ guarded if X is guarded in A.

We embed IFP into CFP by defining for every IFP-formula A a CFP-formula A^{S} (if $\Phi = \lambda X \lambda \vec{x}\,.\, A$, we set $\Phi^{\mathrm{S}} = \lambda X \lambda \vec{x}\,.\, A^{\mathrm{S}}$):

$$\begin{aligned}
A^{\mathrm{S}} &= A \quad \text{if } A \text{ is atomic, i.e. of the form } P(\vec{t}) \text{ or } X(\vec{t}) \\
(A \wedge B)^{\mathrm{S}} &= A^{\mathrm{S}} \wedge B^{\mathrm{S}} \\
(A \to B)^{\mathrm{S}} &= A^{\mathrm{S}} \to B^{\mathrm{S}} \\
(A \vee B)^{\mathrm{S}} &= \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}}) \\
(\forall x\, A)^{\mathrm{S}} &= \forall x\, A^{\mathrm{S}} \\
(\exists x\, A)^{\mathrm{S}} &= \mathrm{S}(\exists x\, A^{\mathrm{S}}) \\
((\nu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\nu\, \Phi^{\mathrm{S}})(\vec{t}) \\
((\mu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\mu\, \Phi^{\mathrm{S}})(\vec{t}) \quad \text{if } \Phi \text{ is guarded} \\
((\mu\, \Phi)(\vec{t}))^{\mathrm{S}} &= (\mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S}))(\vec{t}) \quad \text{if } \Phi \text{ is not guarded}
\end{aligned}$$

The guardedness condition for $\Phi = \lambda X \lambda \vec{a} A$ in $\mu\,\Phi$ is satisfied in all inductive definitions given by two or more "rules" (for example the natural numbers are defined by two rules) since then the predicate variable X occurs only within a disjunction.

▶ **Lemma 13.** CFP *proves* $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$ *(hence* $\mathrm{S}(A^{\mathrm{S}}) \leftrightarrow A^{\mathrm{S}}$*) for all formulas* A *without free predicate variables.*

Proof. See Appendix. ◀

▶ **Theorem 14** (Concurrent embedding)**.** *If* $\Gamma \vdash_{\mathrm{IFP}} A$*, then* $\Gamma^{\mathrm{S}} \vdash_{\mathrm{CFP}} A^{\mathrm{S}}$ *for all formulas* A *without free predicate variables.*

Proof. See Appendix. The proof depends crucially on Lemma 13. ◀

Realizability for CFP is formalized in a system RCFP that extends both CFP and RIFP. We extend the programming language introduced in Sect. 2 by constructs for concurrent computation, where the latter is modeled by a family of computations indexed by a set $\mathcal{I}$ which is the least set containing the constant $*$ and with i, j the elements $\mathrm{L}(i)$, $\mathrm{R}(i)$ and (i, j). We denote elements of $\mathcal{I}$ by *index terms*, denoted $s, t, \ldots$, which are first-order terms built from index variables $\alpha, \beta, \ldots$ and the constant $*$ using the constructors $\mathrm{L}(_)$, $\mathrm{R}(_)$, and $(_,_)$. Closed index terms can be identified with elements of $\mathcal{I}$ and will be called *indices* and denoted $i, j, k, \ldots$. *Concurrent program terms* are defined like the program terms in Sect. 2, but with the extra constructs of *non-deterministic choice* $\langle\alpha\rangle M$ (binding α in M), *index application* $M \cdot s$, and *pattern matching on indices*

$$\text{case } s \text{ of } \{* \to K\,;\, \mathrm{L}(\alpha) \to L\,;\, \mathrm{R}(\alpha) \to M\,;\, (\alpha, \beta) \to N\}$$

(binding α in L and M and α, β in N). In a pattern matching we may omit some of the clauses in which case the omitted clauses have the default value nil. For example, case s of $\{(\alpha, \beta) \to M\}$ stands for case s of $\{* \to \mathrm{nil}\,;\, \mathrm{L}(_) \to \mathrm{nil}\,;\, \mathrm{R}(_) \to \mathrm{nil}\,;\, (\alpha, \beta) \to M\}$. We use the abbreviation $\langle\alpha, \beta\rangle M$ for the term $\lambda\gamma\,.\,\text{case } \gamma \text{ of } \{(\alpha, \beta) \to M\}$ and define, for later use, the terms

$$\begin{aligned} \text{return} &:= \lambda a\, \langle\alpha\rangle \mathrm{inr}(a) \\ \text{bind} &:= \lambda c \lambda f\, \langle\alpha, \beta\rangle \text{case } c \cdot \alpha \text{ of } \{\mathrm{inl}(a) \to (f\, a) \cdot \beta\,;\, \mathrm{inr}(_) \to \mathrm{nil}\} \end{aligned}$$

The logical language of RCFP extends the language of RIFP by a sort of indices and quantification over indices. The specification of programs is extended in RCFP by the equations

$$\begin{aligned} (\langle\alpha\rangle M) \cdot s &= M[s/\alpha] \\ \text{case } * \text{ of } \{\ldots\} &= K \\ \text{case } \mathrm{L}(s) \text{ of } \{\ldots\} &= L[s/\alpha] \\ \text{case } \mathrm{R}(s) \text{ of } \{\ldots\} &= M[s/\alpha] \\ \text{case } (s, t) \text{ of } \{\ldots\} &= N[s, t/\alpha, \beta] \end{aligned}$$

where $\{\ldots\} = \{* \to K\,;\, \mathrm{L}(\alpha) \to L\,;\, \mathrm{R}(\alpha) \to M\,;\, (\alpha, \beta) \to N\}$.

Realizability for formulas of the form $\mathrm{S}(A)$ is defined as

$$c\,\mathbf{r}\,\mathrm{S}(A) \quad = \quad \exists \alpha, a\,(c \cdot \alpha = \mathrm{inl}(a)) \wedge \forall \alpha, a\,(c \cdot \alpha = \mathrm{inl}(a) \to a\,\mathbf{r}\,A)$$

We do *not* stipulate closure of nc or ff formulas under S. Therefore, the sets of nc and ff formulas remains unchanged and the Lemma 1 remains valid for RCFP:

▶ **Lemma 15.** *If A is an ncc formula in* CFP*, then $a \,\mathbf{r}\, A$ is equivalent to A.*

▶ **Theorem 16** (Soundness for CFP)**.** *From a proof in* CFP *of $\Delta, \Gamma \vdash A$, where Δ consists of ncc formulas, one can extract a concurrent program term M such that* RCFP *proves $\Delta, \vec{a} \,\mathbf{r}\, \Gamma \vdash (M\,\vec{a}) \,\mathbf{r}\, A$.*

Proof. It suffices to verify the realizability of the new proof rules (S^+), (S^-), (S^{nc}), that is, to find realizers of the formulas
(a) $A \to \mathrm{S}(A)$,
(b) $\mathrm{S}(A) \to (A \to \mathrm{S}(B)) \to \mathrm{S}(B)$,
(c) $\mathrm{S}(A) \to A$ if A is ncc.
It is easy to see that return realizes (a) and bind realizes (b). We show that $\lambda c\,.\,\mathrm{nil}$ realizes (c). Assume $c \,\mathbf{r}\, \mathrm{S}(A)$. We have to show $\mathrm{nil} \,\mathbf{r}\, A$, that is A, by Lemma 15. Since $c \,\mathbf{r}\, \mathrm{S}(A)$, there exist α and a such that $c \cdot \alpha = \mathrm{inl}(a)$ and $a \,\mathbf{r}\, A$. Hence A, by Lemma 15. ◀

A program *a concurrently realizes* an IFP-formula A, written $a \,\mathbf{cr}\, A$, if a realizes A^{S}, that is, $a \,\mathbf{r}\, A^{\mathrm{S}}$. Combining the Embedding Theorem (Thm. 14), the Soundness Theorem for CFP (16) and the fact that realizability of A^{S} is equivalent to A for non-computational formulas (Lemma 1), one obtains:

▶ **Theorem 17** (Concurrent Soundness)**.** *From a proof of $\Delta, \Gamma \vdash_{\mathrm{IFP}} A$, where Δ is non-computational, one can extract a concurrent program M that maps concurrent realizers of Γ to a concurrent realizer of A, that is $\Delta, \vec{a} \,\mathbf{cr}\, \Gamma \vdash_{\mathrm{CFP}} (M \cdot \vec{a}) \,\mathbf{cr}\, A$.*

In order to understand what kind of extracted program we can expect from the proof of $(\mathrm{G} \subseteq \mathrm{C})^{\mathrm{S}}$, which will be obtained from the proof of $\mathrm{G} \subseteq \mathrm{C}$ (Theorem 12) using Theorem 14 and the realizer of (DP^{S}) (Lemma 30), let us write out this formula:

$$\begin{aligned}
(\mathrm{G} \subseteq \mathrm{C})^{\mathrm{S}} &= \forall x\,.\,\mathrm{G}^{\mathrm{S}}(x) \to \mathrm{C}^{\mathrm{S}}(x) \\
\mathrm{C}^{\mathrm{S}}(x) &\overset{\nu}{=} \mathrm{S}\Big(\bigvee_{d \in \mathrm{SD}} x \in \mathbb{I}_d \wedge \mathrm{C}^{\mathrm{S}}(2x - d)\Big) \\
\mathrm{G}^{\mathrm{S}}(x) &\overset{\nu}{=} (x \neq 0 \to \mathrm{S}(x < 0 \vee x > 0)) \wedge \mathrm{G}^{\mathrm{S}}(\mathrm{t}(x))
\end{aligned}$$

One sees that the predicates C^{S} and G^{S} are almost the same as the original C and G except that the digits of the streams realizing C^{S} or G^{S} can now be computed non-deterministically and concurrently. The extracted program will be able consume and produce such streams.

6 Semantics of concurrent programs and program extraction for CFP

In this section we define an operational big-step semantics for concurrent program terms and show that it fits with a domain-theoretic semantics (Adequacy Theorem 21). Combined with a denotational Soundness Theorem (Theorem 18) and a Faithfulness Theorem (Theorem 20) we obtain that from a proof of a data-formula A (see below) from assumptions Γ one can extract a concurrent program that computes from concurrent realizers of Γ (non-concurrent) data realizing A.

The denotational model of realizers for CFP is the Scott-domain D defined by the recursive domain equation

$$D = 1 + D + D + D \times D + (D \to D) + D^{\mathcal{I}}$$

where 1 is the one-point domain, $+$ denotes the separated sum, $\times$ denotes the topological product, $(D \to D)$ denotes the continuous function space, and $D^{\mathcal{I}}$ denotes the $\mathcal{I}$-fold

topological product of D. Only the last component $D^{\mathcal{I}}$ is new, the rest is as in [1] Sect 5. The components of the sum on the right-hand side of the equation above are embedded into D via the constructors $\mathrm{Nil} : D$, $\mathrm{In}_0, \mathrm{In}_1 : D \to D$, $\mathrm{Pair} : D \times D \to D$, $\mathrm{Fun} : (D \to D) \to D$, $\mathrm{Fam} : D^{\mathcal{I}} \to D$. Every concurrent program term M has an obvious denotation $[\![M]\!]\xi \in D$ in any given environment ξ that maps all its free index variables to elements of $\mathcal{I}$ and all its free object variables to elements of D. For example, $[\![\lambda a\, M]\!]\xi = \mathrm{Fun}(\lambda a' \in D\,.\,[\![M]\!]\xi[a \mapsto a'])$, $[\![\langle\alpha\rangle M]\!]\xi = \mathrm{Fam}(\lambda i \in \mathcal{I}\,.\,[\![M]\!]\xi[\alpha \mapsto i])$, $[\![(M\,N)]\!]\xi = f([\![N]\!]\xi)$ if $[\![M]\!]\xi = \mathrm{Fun}(f)$, and $[\![(M \cdot s)]\!]\xi = \varphi([\![s]\!]\xi)$ if $[\![M]\!]\xi = \mathrm{Fam}(\varphi)$, otherwise these terms have value $\bot$. If M is closed (i.e. has neither free index nor object variables), we omit the environment.

▶ **Theorem 18** (Denotational soundness). *If* RCFP *proves* $\Gamma \vdash B$*, then* B *holds in every model of* Γ *that interprets the sort of realizers as* D*. In particular, if* B *is of the form* $M\,\mathbf{r}\,A$*, then* $[\![M]\!] \in D$ *realizes* A *in that model.*

We call a CFP-formula a *parametric data formula* if every subformula of the form $A \to B$ or $\nu\,\Phi(t)$ is non-computational. A data formula is a parametric data formula without free predicate variables. Furthermore, *data terms* are defined, as in [1], as the terms built from Nil by injections and pairing. As in [1] we identify data terms with the corresponding elements in D and call them simply *data*

Our main result is analogous to the Program Extraction Theorem in [1] and refers to a big-step operational semantics. First we introduce *closures* which are inductively defined as pairs (M, η) where M is a term and η is a finite mapping from object variables to closures. A *value* is a closure (M, η) where M is an *intro term*, that is, either Nil or an injection or a pair or a λ-abstraction or a choice term, $\langle\alpha\rangle M$. The *bigstep reduction relation* $c \longrightarrow v$ between closures c and values v is inductively defined as in [1], but with additional five rules:

$$\frac{(M,\eta) \longrightarrow (\langle\alpha\rangle M_0, \eta') \qquad (M_0[s/\alpha], \eta') \longrightarrow v}{(M\,s, \eta) \longrightarrow v}$$

In the remaining four rules we use the abbreviation

$$\vec{C} := * \to M_* \,;\, \mathrm{L}(\alpha) \to M_0 \,;\, \mathrm{R}(\alpha) \to M_1 \,;\, (\alpha, \beta) \to M$$

$$\frac{(M_*, \eta) \longrightarrow v}{(\text{case } *\text{ of } \{\vec{C}\}, \eta) \longrightarrow v} \qquad \frac{(M_0[s/\alpha_p], \eta) \longrightarrow v}{(\text{case } \mathrm{L}(s) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v} \qquad \frac{(M_1[s/\alpha_p], \eta) \longrightarrow v}{(\text{case } \mathrm{R}(s) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v}$$

$$\frac{(M[s,t/\alpha,\beta], \eta) \longrightarrow v}{(\text{case } (s,t) \text{ of } \{\vec{C}\}, \eta) \longrightarrow v}$$

Note that neither the denotational semantics nor the bigstep operational semantics exhibit any kind of non-determinism or concurrency. These features come into play only through the *printing relation*, $c \Longrightarrow d$, between closures c and data d, defined below. It is inductively defined as in [1], but with five additional rules concerned with non-deterministic choice:

$$\frac{c \longrightarrow (\langle\alpha\rangle M, \eta) \qquad (M, \eta) \Longrightarrow d}{c \Longrightarrow d}$$

In the remaining four rules (where $p = 0, 1$) α must occur free in c and β must be fresh:

$$\frac{c[*/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[\mathrm{L}(\alpha)/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[\mathrm{R}(\alpha)/\alpha] \Longrightarrow d}{c \Longrightarrow d} \qquad \frac{c[(\alpha,\beta)/\alpha] \Longrightarrow d}{c \Longrightarrow d}$$

If M is a closed term, then we write $M \Longrightarrow d$ instead of $(M, \emptyset) \Longrightarrow$ where $\emptyset$ is the empty mapping. Essentially, these rules allow to reduce a closure c with a choice parameter α to $c[i/\alpha]$ for any $i \in \mathcal{I}$. The way the rules are set up, one can do the instantiation $c[i/\alpha]$ lazily and incrementally, by only specifying the outer shape of i in one step.

▶ **Theorem 19** (Program Extraction). *From a proof of a data formula A in* CFP *from concurrently realizable assumptions one can extract a concurrent program term M such $M \Longrightarrow d$ for some data d provably realizing A.*

The main building blocks for the proof of the Program Extraction Theorem are the Soundness Theorem (Theorem 16), the Computational Adequacy Theorem and the Faithfulness Theorem below. The latter two refer to the relation $d \in \mathrm{data}(a)$, for $a \in D$ and data d, which is defined inductively as follows:

(i) $\mathrm{Nil} \in \mathrm{data}(\mathrm{Nil})$.
(ii) If $d \in \mathrm{data}(a)$, then $\mathrm{In}_p\, d \in \mathrm{data}(\mathrm{In}_p\, a)$ for $p = 0, 1$.
(iii) If $d_p \in \mathrm{data}(a_p)$ for $p = 0, 1$, then $\mathrm{Pair}\, d_1\, d_1 \in \mathrm{data}(\mathrm{Pair}\, a_0\, a_1)$.
(iv) If $d \in \mathrm{data}(\varphi\, i)$ for some $i \in \mathcal{I}$, then $d \in \mathrm{data}(\mathrm{Fam}\, \varphi)$.

▶ **Theorem 20** (Faithfulness). *If $a \in D$ concurrently realizes a data formula A, then* $\mathrm{data}(a)$ *is nonempty and all $d \in \mathrm{data}(a)$ realize A, provably in* IFP.

Proof. See Appendix. ◀

▶ **Theorem 21** (Computational Adequacy). *If $d \in \mathrm{data}([\![M]\!])$, then $M \Longrightarrow d$.*

The proof of Computational Adequacy is quite involved and will occupy the rest of this section.

Proof of the Program Extraction Theorem from Soundness, Computational Adequacy and Faithfulness. From a proof of a data formula A in CFP from realizable assumptions, one obtains by Soundness a concurrent program term M realizing A. More precisely, $[\![M]\!]$ realizes A. By Faithfulness, there is a data $d \in \mathrm{data}([\![M]\!])$ such that d provably deterministically realizes A. By Computational Adequacy, $M \Longrightarrow d$. ◀

Now we develop the necessary machinery to prove Computational Adequacy. For a closure c we let $\overline{c}$ be the closed term represented by c, that is,

$$\overline{(M, \eta)} = M[\overline{\eta(x)}/x \mid x \in \mathrm{FV}(M)].$$

The proof is done through the following series of lemmas whose proofs can be found in the Appendix.

▶ **Lemma 22** (Correctness).
(a) *If $c \longrightarrow v$, then $\vdash \overline{c} = \overline{v}$.*
(b) *If $c \Longrightarrow d$, then $\vdash d \in \mathrm{data}([\![\overline{c}]\!])$.*

▶ **Lemma 23** (Instantiation).
(a) *If $c[i/\alpha] \Longrightarrow d$, then $c \Longrightarrow d$.*
(b) *If $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \Longrightarrow d$, then $c \Longrightarrow d$.*

▶ **Lemma 24** (Irreducibility of values). *For values v, v'*

$$v \longrightarrow v' \quad \textit{iff} \quad v = v'\,.$$

Let D_0 be the set of compact elements of D. Every $a \in D_0$ has a natural *rank*, $\mathbf{rk}(a) \in \mathbb{N}$, satisfying properties $\mathbf{rk}1 - \mathbf{rk}4$. The first three properties are as in [1], the fourth is
rk4 If $\mathrm{Fam}\, \varphi$ is compact, then for every $i \in \mathcal{I}$, $\varphi\, i$ is compact with $\mathbf{rk}(\varphi\, i) < \mathbf{rk}(\mathrm{Fam}\, \varphi)$.

To every $a \in D_0$ we assign a set of closures $\mathrm{Cl}(a)$ by recursion on $\mathbf{rk}(a)$. The definition is as in [1], with the extra clause

$$\mathrm{Cl}(\mathrm{Fam}\,\varphi) = \{c \mid \exists \alpha, M, \eta\,.\, c \longrightarrow (\langle\alpha\rangle M, \eta) \wedge \forall i \in \mathcal{I}\,.\,(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)\}$$

A similar assignment of closures to the compact elements of a semantic domain is used in [15].

▶ **Lemma 25** (Monotonicity). *If a, b are compact elements in D such that $a \sqsubseteq b$, then* $\mathrm{Cl}(a) \supseteq \mathrm{Cl}(b)$.

▶ **Lemma 26** (Printing of data). *If $c \in \mathrm{Cl}(a)$ and $d \in \mathrm{data}(a)$, then $c \Longrightarrow d$.*

▶ **Lemma 27** (Reducibility of closures). *$c \in \mathrm{Cl}(a)$ iff $c \longrightarrow v$ for some $v \in \mathrm{Cl}(a)$.*

We write $\eta \in \mathrm{Cl}(\xi)$ if η and ξ have the same domain and $\eta(x) \in \mathrm{Cl}(\xi(x))$ for every object variable x and $\eta(\alpha) = \xi(\alpha)$ for every index variable α in the common domain.

▶ **Lemma 28** (Approximation). *If $\eta \in \mathrm{Cl}(\xi)$, $a \in D_0$ and $a \sqsubseteq [\![M]\!]\xi$, then $(M, \eta) \in \mathrm{Cl}(a)$.*

▶ **Lemma 29.** *If $d \in \mathrm{data}(a)$, then $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq a$.*

Proof of the Adequacy Theorem (Theorem 21). Assume $d \in \mathrm{data}([\![M]\!])$ where M is closed. By Lemma 29, $d \in \mathrm{data}(a)$ for some compact $a \sqsubseteq [\![M]\!]$. By Lemma 28, $(M, \emptyset) \in \mathrm{Cl}(a)$. By Lemma 26, $M \Longrightarrow d$. ◀

7 Realizing the Disjunction Principle

In this Section we show that the Disjunction Principle can be concurrently realized.

Recall that in Theorem 12 we proved in IFP that $\mathrm{G} \subseteq \mathrm{C}$, with the help of (DP). By Concurrent Soundness (Theorem 16), we can extract from this proof a concurrent program transforming infinite Gray code into signed digit representation, provided we can concurrently realize (DP).

▶ **Lemma 30.** *The Disjunction Principle can be concurrently realized.*

Proof. The embedding of the Disjunction Principle, $(\mathrm{DP})^{\mathrm{S}}$, is

$$(A \overset{P}{\vee} B)^{\mathrm{S}} \wedge (P \overset{Q}{\vee} C)^{\mathrm{S}} \to (A \vee B \vee C)^{\mathrm{S}}$$

where $(A \overset{P}{\vee} B)^{\mathrm{S}} = (P \to \mathrm{S}(A \vee B)) \wedge (\neg P \to A \wedge B)$, $(P \overset{Q}{\vee} C)^{\mathrm{S}} = (Q \to \mathrm{S}(P \vee C)) \wedge (\neg Q \to P \wedge C)$ and $(A \vee B \vee C)^{\mathrm{S}} = \mathrm{S}(\mathrm{S}(A \vee B) \vee C)$. Since CFP proves that $\mathrm{S}(\mathrm{S}(A \vee B) \vee C)$ is equivalent to $\mathrm{S}((A \vee B) \vee C)$ (easy exercise), it suffices to realize the formula

$$(*) \qquad (A \overset{P}{\vee} B)^{\mathrm{S}} \wedge (P \overset{Q}{\vee} C)^{\mathrm{S}} \to \mathrm{S}((A \vee B) \vee C)$$

where P, Q, A, B, C are ncc formulas. The following program realizes $(*)$: $f_{\mathrm{DP}} = \lambda c\,.$

$$\begin{aligned}\langle\gamma\rangle\,.\,\mathrm{case}\ \gamma\ \mathrm{of}\quad & \{\mathrm{L}(\alpha) \to \mathrm{case}\ \pi_1(c)\cdot\alpha\ \mathrm{of}\ \{\mathrm{inl}(\mathrm{inl}(_)) \to \mathrm{inl}(\mathrm{inl}(\mathrm{inl}(\mathrm{nil})))\,;\\ & \qquad\qquad \mathrm{inl}(\mathrm{inr}(_)) \to \mathrm{inl}(\mathrm{inl}(\mathrm{inr}(\mathrm{nil})))\}\,;\\ & \mathrm{R}(\beta) \to \mathrm{case}\ \pi_2(c)\cdot\beta\ \mathrm{of}\ \{\mathrm{inl}(\mathrm{inr}(_)) \to \mathrm{inl}(\mathrm{inr}(\mathrm{nil}))\}\}\end{aligned}$$

In order to show that f_{DP} realizes $(*)$, we assume $a := \pi_1(c)$ realizes $(A \overset{P}{\vee} B)^{\mathrm{S}}$ and $b := \pi_2(c)$ realizes $(P \overset{Q}{\vee} C)^{\mathrm{S}}$. We show that $f_{\mathrm{DP}}\, c$ realizes $\mathrm{S}((A \vee B) \vee C)$. The assumptions mean:

(1) If P, then a concurrently realizes $A \vee B$, i.e.
  (1.1) $a \cdot i_0 = \mathrm{inl}(a_0)$ for some i_0, a_0,
  (1.2) If $a \cdot i = \mathrm{inl}(a')$, then $a' \,\mathbf{r}\, (A \vee B)$, that is, $a' = \mathrm{inl}(_)$ and A, or $a' = \mathrm{inr}(_)$ and B.
(2) If Q, then b concurrently realizes $P \vee C$, i.e.
  (2.1) $b \cdot j_0 = \mathrm{inl}(b_0)$ for some j_0, b_0,
  (2.2) If $b \cdot j = \mathrm{inl}(b')$, then $b' \,\mathbf{r}\, (P \vee C)$, that is, $b' = \mathrm{inl}(_)$ and P, or $b' = \mathrm{inr}(_)$ and C.
(3) If $\neg P$, then A and B.
(4) If $\neg Q$, then P and C

In the proof that $f_{\mathrm{DP}}(a, b)$ realizes $\mathrm{S}((A \vee B) \vee C)$ we argue classically, admitting case analysis on P and Q.

The first condition holds since, if P holds, then, by (1), $f_{\mathrm{DP}}(a, b) \cdot \mathrm{L}(i_0)$ is of the form $\mathrm{inl}(_)$. If P does not hold, then Q holds, by (4), and therefore $b \cdot j_0 = \mathrm{inl}(b_0)$, by (2.1). By (2.2), $b_0 = \mathrm{inr}(_)$, since P does not hold. Hence $f_{\mathrm{DP}}(a, b) \cdot \mathrm{R}(j_0)$ is of the form $\mathrm{inl}(_)$.

To verify the second condition, assume $f_{\mathrm{DP}}(a, b) \cdot k = \mathrm{inl}(c)$. We have to show that c realizes $(A \vee B) \vee C$. By the definition of $f_{\mathrm{DP}}(a, b)$, k is of the form $\mathrm{L}(i)$ or $\mathrm{R}(j)$.

If $k = \mathrm{L}(i)$, then either $a \cdot i = \mathrm{inl}(\mathrm{inl}(_))$ and $c = \mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$, or else $a \cdot i = \mathrm{inl}(\mathrm{inr}(_))$ and $c = \mathrm{inl}(\mathrm{inr}(\mathrm{nil}))$. If P holds, then a concurrently realizes $A \vee B$, by (1). Hence, if $a \cdot i = \mathrm{inl}(\mathrm{inl}(_))$, then A holds, hence $\mathrm{inl}(\mathrm{nil})$ realizes $A \vee B$ and consequently $c = \mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$ realizes $(A \vee B) \vee C$. The case that $a \cdot i = \mathrm{inl}(\mathrm{inr}(_))$ is similar. If P does not hold, then A and B hold, by (3), hence $\mathrm{inl}(\mathrm{nil})$ and $\mathrm{inr}(\mathrm{nil})$ both realize $A \vee B$. It follows in any case that c realizes $(A \vee B) \vee C$.

If $k = \mathrm{R}(j)$, then $b \cdot j = \mathrm{inl}(\mathrm{inr}(_))$ and $c = \mathrm{inr}(\mathrm{nil})$. If Q holds then C holds by (2.2). If Q does not hold, then C holds by (4). In either case c realizes $((A \vee B) \vee C$. ◀

In order to understand f_{DP} we express its behaviors in terms of overlapping defining equations that ignore the indices in $(\in \mathcal{I})$ labeling the different choices. Ignoring also the leading $\mathrm{inl}(\cdot)$ of the inputs and outputs, which only flag up a valid result, we obtain (writing `(a,b)` for $\langle a, b\rangle$)

```
fDP (inl(_),b) = inl(inl(nil))
fDP (inr(_),b) = inl(inr(nil))
fDP (a,inr(_)) = inr(nil)
```

These equation must be interpreted as non-deterministic rewrite rules, similar to the program `gtos` in the introduction.

8 Extracting programs for infinite Gray code

We conclude by extracting the programs from the proofs of the Lemmas 6-11 in Sect. 4, and assembling them, via program extraction from the proof of Theorem 12, to yield the main program transforming infinite Gray to signed digit representation. Since the proofs are so short it is easy to read off the extracted programs from the proof "by hand".

For signed digit streams (that is, realizers of $\mathrm{C}(x)$) we display the possible digits $\mathrm{inl}(\mathrm{inl}(\mathrm{nil}))$, $\mathrm{inl}(\mathrm{inr}(\mathrm{nil}))$, $\mathrm{inr}(\mathrm{nil})$ as `-1,1,0`, respectively. For infinite Gray codes (that is, realizers of $\mathrm{G}(x)$) we display the possible digits $\mathrm{inl}(\mathrm{nil})$, $\mathrm{inr}(\mathrm{nil})$ as `0,1`, respectively. Note that with this display the program `fDP` (which will be used in the next program) reads

```
fDP (0,b) = -1
fDP (1,b) = 1
fDP (a,1) = 0
```

Furthermore, we display nested pairs like $\langle a, \langle b, s\rangle\rangle$ as `a:b:s`.

Lemma 6. If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

```
f6 (a:b:s) = fDP (a,b)
```

Lemma 7. (a) If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$. (b) If $x \in \mathrm{G}$, then $|x| \in \mathrm{G}$.

```
f7a (a:s) = swap a : s    where {swap 0 = 1;  swap 1 = 0}
f7b (a:s) = 1:s
```

Lemma 8. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x-1)$.

```
f8 (a:s) = f7a s
```

Lemma 9. If $\mathrm{G}(x)$, then $\mathrm{G}(\mathrm{t}(x)$.

```
f9 (a:s) = s
```

Lemma 10. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1-x)$.

```
f10 s = 1 : f8 s
```

Lemma 11. If $-\frac{1}{2} \leq x \leq \frac{1}{2}$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x)$.

```
f11 (a:s) = a : f9 (f10 s)
```

Theorem 12. $\mathrm{G} \subseteq \mathrm{C}$.

```
f12 s = let { d = f6 s } in
   d : case d of {-1 -> f12 (f9 s); 0 -> f12 (f11 s); 1 -> f12 (f8 s)}
```

Hence

```
f12 (0:s) = -1 : f12 s
f12 (1:a:s) = 1 : f12 (swap a : s)
f12 (a:1:c:s) = 0 : f12 (a : swap c : s)
```

Again, the equations above should be read as overlapping rewrite rules. Observe that the equations for `f12` correspond exactly to the equations given for the program `gtos` shown in the introduction.

9 Conclusion

We introduced a logic and realizability interpretation for the extraction of non-deterministic concurrent programs and applied it to extract Tsuiki's program converting infinite Gray code for real numbers into signed digit representation. Through the Soundness and Computational Adequacy Theorems, extracted programs come with formal proofs of their correctness and termination.

Although we are still far from a fully fledged method of certified code generation for non-deterministic and concurrent programs, we believe that our application to infinite Gray code (which was done on paper) can be viewed as a proof of concept that makes it worthwhile to implement this method in a suitable proof system.

Regarding further applications, computable analysis holds plenty of other inherently non-deterministic problems (for example, root finding or inversion of matrices with real valued entries) to which our method can be applied.

References

1 U. Berger. Realisability for induction and coinduction with applications to constructive analysis. *Jour. Universal Comput. Sci.*, 16(18):2535–2555, 2010.

2 U. Berger and T. Hou. A realizability interpretation of Church's simple theory of types. *Mathematical Structures in Computer Science*, 2016. To appear.

3 U. Berger, K. Miyamoto, H. Schwichtenberg, and M. Seisenberger. Minlog – a tool for program extraction for supporting algebra and coalgebra. In *CALCO-Tools*, volume 6859 of *LNCS*, pages 393–399. Springer Verlag, Berlin, Heidelberg, New York, 2011. doi:10.1007/978-3-642-22944-2_29.

4 U. Berger, K. Miyamoto, H. Schwichtenberg, and H. Tsuiki. Logic for Gray-code computation. In *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, Ontos Mathematical Logic 6. de Gruyter, 2016. To appear.

5 S. Berghofer. Program Extraction in simply-typed Higher Order Logic. In *Types for Proofs and Programs (TYPES'02)*, volume 2646 of *LNCS*, pages 21–38. Springer Verlag, Berlin, Heidelberg, New York, 2003.

6 A. Bucciarelli, T. Ehrhard, and G. Manzonetto. A relational semantics for parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7):918–934, 2012.

7 C. M. Chuang. *Extraction of Programs for Exact Real Number Computation Using Agda*. PhD thesis, Swansea University, 2011.

8 R.L. Constable. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice–Hall, New Jersey, 1986.

9 S. Hayashi and H. Nakano. *PX: A Computational Logic*. MIT Press, 1988.

10 Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

11 C.-H.L. Ong. Non-determinism in a functional setting. In *Proc. of LICS'93*, pages 275–286, 1993.

12 C. Paulin-Mohring. Inductive definitions in the system Coq; rules and properties. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, pages 328–345. LNCS Vol. 664, 1993.

13 G.D. Plotkin. A powerdomain construction. *SIAM J. Comput.*, 5(3):452–487, 1976.

14 H. Tsuiki. Real Number Computation through Gray Code Embedding. *Theoretical Computer Science*, 284(2):467–485, 2002.

15 G. Winskel. *The Formal Semantics of Programming Languages*. Foundations of Computing Series. The MIT Press, Cambridge, Massachusetts, 1993.

A Appendix: Proofs

Lemma 1. If A is an ncc formula, then $a\,\mathbf{r}\,A$ is equivalent to A, provably in RIFP.

Proof. For a predicate $\mathcal{P}$ let $\mathrm{K}\,\mathcal{P} = \lambda(a,\vec{x})\,\mathcal{P}(\vec{x})$ (that is, $\mathcal{P}$ is extended by an extra dummy first argument ranging over realizers). For a predicate $\mathcal{Q}$ with first argument ranging over realizers let $\exists\,\mathcal{Q} = \lambda\vec{x}\,\exists a\,\mathcal{Q}(a,\vec{x})$ and $\forall\,\mathcal{Q} = \lambda\vec{x}\,\forall a\,\mathcal{Q}(a,\vec{x})$. Note that $\exists$ is left adjoint to K, that is $\exists\,\mathcal{Q} \subseteq \mathcal{P}$ iff $\mathcal{Q} \subseteq \mathrm{K}\,\mathcal{P}$, and $\forall$ is right adjoint to K, that is $\mathcal{P} \subseteq \forall\,\mathcal{Q}$ iff $\mathrm{K}\,\mathcal{P} \subseteq \mathcal{Q}$. Hence $\mathrm{K}\,(\forall\,\mathcal{Q}) \subseteq \mathcal{Q} \subseteq \mathrm{K}\,(\exists\,\mathcal{Q})$. In addition, if $\mathcal{Q} = \mathrm{K}\,\mathcal{P}$, then $\forall\,\mathcal{Q} = \exists\,\mathcal{Q} = \mathcal{P}$. For an IFP-formula A, let

$$\begin{aligned} \mathrm{K}_A &= \{\tilde{X} = \mathrm{K}\,X \mid X \text{ free in } A\} \\ \exists_A &= \{\exists\,\tilde{X} = X \mid X \text{ free in } A\} \end{aligned}$$

We show, more generally, that RIFP proves

(a) $\mathrm{K}_A \vdash A \leftrightarrow a\,\mathbf{r}\,A$ for nc formulas A.
(b) $\exists_A \vdash A \leftrightarrow \exists a\,a\,\mathbf{r}\,A$ for ff formulas A.
The proof is by simultaneous induction on A.

For (a), the only non-obvious cases are implication as well as inductive and coinductive definitions.

Consider $A \to B$ where A is ffc and B is nc. W.l.o.g. let us assume that A is not ncc. Hence $c\,\mathbf{r}\,(A \to B)$ is $\forall a(a\,\mathbf{r}\,A \to (c \cdot a)\,\mathbf{r}\,B)$. By induction hypothesis (a) this is equivalent to $\forall a(a\,\mathbf{r}\,A \to B)$ and hence, by induction hypothesis (b) and since $\mathrm{K}_A \vdash \exists_A$, to $A \to B$.

Consider $(\mu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $(\mu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\mu\,\Phi)(\vec{t})$. We show more generally $\mu\,\mathbf{r}(\Phi) = \mathrm{K}\,(\mu\,\Phi)$. We show the inclusion $\mu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\mu\,\Phi)$ by induction. Hence, we have to show $\mathbf{r}(\Phi)(\mathrm{K}\,(\mu\,\Phi)) \subseteq \mathrm{K}\,(\mu\,\Phi)$, that is, $a\,\mathbf{r}\,A \to A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\mu\,\Phi)$ and $X = \mu\,\Phi$. But the extra assumptions imply $\tilde{X} = \mathrm{K}\,X$. Hence, induction hypothesis (a) applies. The other inclusion, $\mathrm{K}\,(\mu\,\Phi) \subseteq \mu\,\mathbf{r}(\Phi)$, is equivalent to $\mu\,\Phi \subseteq \forall(\mu\,\mathbf{r}(\Phi))$. We show $\mu\,\Phi \subseteq \forall\,(\mu\,\mathbf{r}(\Phi))$ by induction. Hence, we have to show $\Phi(\forall\,(\mu\,\mathbf{r}(\Phi))) \subseteq \forall\,(\mu\,\mathbf{r}(\Phi))$, that is, $A \to a\,\mathbf{r}\,A$ under the extra assumptions $X = \forall\,(\mu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mu\,\mathbf{r}(\Phi)$. Since $\mathrm{K}\,(\forall\,(\mu\,\mathbf{r}(\Phi))) \subseteq \mu\,\mathbf{r}(\Phi)$ and $a\,\mathbf{r}\,A$ is monotone in $\tilde{X}$ (and A is independent of $\tilde{X}$), it suffices to show that $A \to a\,\mathbf{r}\,A$ follows form the extra assumptions $X = \forall\,(\mu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mathrm{K}\,(\forall\,(\mu\,\mathbf{r}(\Phi)))$. But this is guaranteed by induction hypothesis (a).

For coinduction, the proof is obtained by dualization, that is, by inverting all inclusions and implications and replacing μ, "induction", $\forall$ by ν, "coinduction", $\exists$, respectively. More precisely, consider $(\nu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $(\nu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\nu\,\Phi)(\vec{t})$. We show more generally $\nu\,\mathbf{r}(\Phi) = \mathrm{K}\,(\nu\,\Phi)$. We show the inclusion $\mathrm{K}\,(\nu\,\Phi) \subseteq \nu\,\mathbf{r}(\Phi)$ by coinduction. Hence, we have to show $\mathrm{K}\,(\nu\,\Phi) \subseteq \mathbf{r}(\Phi)(\mathrm{K}\,(\nu\,\Phi))$, that is, $A \to a\,\mathbf{r}\,A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\nu\,\Phi)$ and $X = \nu\,\Phi$, which holds by induction hypothesis (a). The other inclusion, $\nu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\nu\,\Phi)$, is equivalent to $\exists(\nu\,\mathbf{r}(\Phi)) \subseteq \nu\,\Phi$, where for a predicate $\mathcal{P}$, $\exists\,\mathcal{P} = \lambda \vec{x}\,\exists a\;\mathcal{P}(a, \vec{x})$. We show $\exists(\nu\,\mathbf{r}(\Phi)) \subseteq \nu\,\Phi$ by coinduction. Hence, we have to show $\exists\,(\nu\,\mathbf{r}(\Phi)) \subseteq \Phi(\exists\,(\nu\,\mathbf{r}(\Phi)))$, that is, $a\,\mathbf{r}\,A \to A$ under the extra assumptions $X = \exists\,(\nu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \nu\,\mathbf{r}(\Phi)$. Since $\nu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\exists\,(\mu\,\mathbf{r}(\Phi)))$ and $a\,\mathbf{r}\,A$ is monotone in $\tilde{X}$, it suffices to show that $a\,\mathbf{r}\,A \to A$ follows form the extra assumptions $X = \exists\,(\nu\,\mathbf{r}(\Phi))$ and $\tilde{X} = \mathrm{K}\,(\exists\,(\nu\,\mathbf{r}(\Phi)))$. But this is guaranteed by induction hypothesis (a).

For (b), the only non-obvious case is induction.

Consider $(\mu\,\Phi)(\vec{t})$ where $\Phi = \lambda X \lambda \vec{x}\,A$. We have to show $\exists a\,(\mu\,\mathbf{r}(\Phi))(a, \vec{t}) \leftrightarrow (\mu\,\Phi)(\vec{t})$. We show more generally $\exists\,(\mu\,\mathbf{r}(\Phi)) = \mu\,\Phi$. The inclusion $\exists\,(\mu\,\mathbf{r}(\Phi)) \subseteq \mu\,\Phi$ is equivalent to $\mu\,\mathbf{r}(\Phi) \subseteq \mathrm{K}\,(\mu\,\Phi)$. We use induction. Hence, we have to show $\mathbf{r}(\Phi)(\mathrm{K}\,(\mu\,\Phi)) \subseteq \mathrm{K}\,(\mu\,\Phi)$, that is, $\exists a\,a\,\mathbf{r}\,A \to A$ under the extra assumptions that $\tilde{X} = \mathrm{K}\,(\mu\,\Phi)$ and $X = \mu\,\Phi$. But the extra assumptions imply $\tilde{X} = \mathrm{K}\,X$ and hence $\exists\,\tilde{X} = X$. Hence, induction hypothesis (b) applies. The other inclusion, $\mu\,\Phi \subseteq \mu\,\exists\,(\mu\,\mathbf{r}(\Phi))$, can be shown by induction. Hence, we show $\Phi(\exists\,(\mu\,\mathbf{r}(\Phi))) \subseteq \exists\,(\mu\,\mathbf{r}(\Phi))$. Since $\exists\,(\mu\,\mathbf{r}(\Phi)) = \exists\,(\mathbf{r}(\Phi)(\mu\,\mathbf{r}(\Phi)))$, this equivalent to $X = \exists\,(\mu\,\mathbf{r}(\Phi)), \tilde{X} = \mu\,\mathbf{r}(\Phi) \vdash A \to \exists a\,a\,\mathbf{r}\,A$. Since the assumptions in this sequent imply $X = \exists\,\tilde{X}$, this is implied by induction hypothesis (b). ◀

Lemma 3
(a) If $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ for some $a, b \in \mathbb{Q}$, then $y \in \mathrm{A}$.
(b) If $x \in \mathrm{C}$ and $b \in \mathbb{Q}$ is such that $x + b \in \mathbb{I}$, then $x + b \in \mathrm{C}$.
(c) If $x \in \mathrm{C}$ and $2x + b \in \mathbb{I}$, where $b \in \mathbb{Q}$, then $2x + b \in \mathrm{C}$.
(d) If $x \in \mathrm{C}$, then $-x \in \mathrm{C}$.

Proof.

(a) Let $x \in \mathrm{A}$ and $y \in \mathbb{I}$ with $y = ax + b$ where $a, b \in \mathbb{Q}$. Let $n \in \mathbb{N}$. Let $k \in \mathbb{N}$ such that $|a| \leq 2^k$ Since $x \in A$ there is $q \in \mathbb{Q}$ such that $|x - q| \leq 2^{-(n+k)}$. Hence

$$|ax + b - (aq + b)| = |a||x - q| \leq 2^k / 2^{n+k} = 2^{-n}$$

Let $q' = -1$ if $aq + b < -1$, $= 1$ if $aq + b > 1$, and $= aq + b$ otherwise. Then $q' \in \mathbb{Q} \cap \mathbb{I}$ and $|ax + b - q'| \leq 2^{-n}$.

(b) Define $P := \{x \in \mathbb{I} \mid \exists b \in \mathbb{Q}\,.\, x + b \in \mathrm{C}\}$. We show $P \subseteq C$ by coinduction. Let $x \in P$, that is $x \in \mathbb{I}$ and $x + b \in \mathrm{C}$ for some $b \in \mathbb{Q}$. We have to find $d \in \mathrm{SD}$ such that $x \in \mathbb{I}_d$ and $2x - d \in P$. Since $x + b \in \mathrm{C}$ we find $d_0, d_1 \in \mathrm{SD}$ such that $2(x + b) - d_0 \in \mathrm{C}$ and $4(x + b) - 2d_0 - d_1 \in \mathrm{C}$. Hence $|4(x + b) - 2d_0 - d_1| \leq 1$, i.e. $|x - a| \leq 1/4$ where $a := d_0/2 + d_1/4 - b$. Therefore $x \in \mathbb{I} \cap [a - 1/4, a + 1/4]$. Choose $d \in \mathrm{SD}$ such that $\mathbb{I} \cap [a - 1/4, a + 1/4] \subseteq \mathbb{I}_d$. Then $x \in \mathbb{I}_d$. With $c := 2b + d - d_0 \in \mathbb{Q}$ we have $2x - d + c = 2(x + b) - d_0 \in \mathrm{C}$, hence $2x - d \in P$.

(c) If $x \in \mathrm{C}$, then $2x - e \in \mathrm{C}$ for some $e \in \mathrm{SD}$. Hence $2x + b \in \mathrm{C}$, by part (b), provided $2x + b \in \mathbb{I}$.

(d) Let $P(x) = -x \in \mathrm{C}$. We show $P \subseteq \mathrm{C}$, by coinduction. Assume $-x \in \mathrm{C}$. Let $d \in \mathrm{SD}$ such that $-x \in \mathbb{I}_d$ and $2(-x) - d \in \mathrm{C}$. Then $x \in \mathbb{I}_{-d}$ and $-(2x - (-d)) \in \mathrm{C}$, i.e. $P(2x - (-d)$. ◀

Lemma 13. CFP proves $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$ (hence $\mathrm{S}(A^{\mathrm{S}}) \leftrightarrow A^{\mathrm{S}}$) for all formulas A without free predicate variables.

Proof. For a formula A we set $\Gamma_A = \{\mathrm{S}(X) \subseteq X \mid X \text{ is not guarded in } A\}$. Note that if A has no free predicate variables, then $\Gamma_A = \emptyset$.

We show more generally $\Gamma_A \vdash_{\mathrm{CFP}} \mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$, by induction on A.

The *cases where* A^{S} *is of the form* $\mathrm{S}(\ldots)$, that is, $A \vee B$ and $\exists x\, A$, are trivial since the modality is idempotent.

For the *case* $P(\vec{t})$ the assertion holds by the rule $(\mathrm{S}^{\mathrm{nc}})$.

For the *case* $X(\vec{t})$ the assertion holds since X is not guarded in $X(\vec{t})$.

Cases $A \wedge B$ *and* $A \to B$. First note that $\Gamma_{A \wedge B} = \Gamma_{A \to B} = \Gamma_A \cup \Gamma_B$. $\mathrm{S}((A \wedge B)^{\mathrm{S}})$ is $\mathrm{S}(A^{\mathrm{S}} \wedge B^{\mathrm{S}})$, which implies $\mathrm{S}(A^{\mathrm{S}}) \wedge \mathrm{S}(B^{\mathrm{S}})$. By induction hypothesis, this is equivalent to $A^{\mathrm{S}} \wedge B^{\mathrm{S}}$, i.e. $(A \wedge B)^{\mathrm{S}}$. For implication the argument is similar.

Cases $\forall x\, A$. $\mathrm{S}((\forall x\, A)^{\mathrm{S}})$ is $\mathrm{S}(\forall x\, A^{\mathrm{S}})$, which implies $\forall x\, \mathrm{S}(A^{\mathrm{S}})$. By induction hypothesis and the rule (S^{+}), this is equivalent to $\forall x\, A^{\mathrm{S}}$, i.e. $(\forall x\, A)^{\mathrm{S}}$.

Case $(\nu\, \Phi)(\vec{t})$. It suffices to show $\mathrm{S}(\nu\, \Phi^{\mathrm{S}}) \subseteq \nu\, \Phi^{\mathrm{S}}$. Define $\mathcal{P} := \nu\, (\Phi^{\mathrm{S}} \circ S)$. Assuming $\Phi = \lambda X \lambda \vec{x}\,.\, A$, we have $\Gamma_A \subseteq \Gamma_{\nu\, \Phi} \cup \{\mathrm{S}(X) \subseteq X\}$. Hence, by induction hypothesis, $\Gamma_{\nu\, \Phi}, \mathrm{S}(X) \subseteq X \vdash_{\mathrm{CFP}} \mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$. In the following, we reason in CFP and assume $\Gamma_{\nu\, \Phi}$. Since $\mathrm{S}(\mathrm{S}(X)) \subseteq \mathrm{S}(X)$, it follows $\mathrm{S}(A^{\mathrm{S}}[\mathrm{S}(X)/X]) \to A^{\mathrm{S}}[\mathrm{S}(X)/X]$, i.e. $\mathrm{S}((\Phi^{\mathrm{S}} \circ \mathrm{S})(X)) \subseteq (\Phi^{\mathrm{S}} \circ \mathrm{S})(X)$. In particular, for $X := \mathcal{P}$ we obtain $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$, since $(\Phi^{\mathrm{S}} \circ \mathrm{S})(\mathcal{P}) = \mathcal{P}$. Therefore, it suffices to show that $\nu\, \Phi^{\mathrm{S}} = \mathcal{P}$. By (S^{+}), $\Phi^{\mathrm{S}} \subseteq \Phi^{\mathrm{S}} \circ S$, hence $\nu\, \Phi^{\mathrm{S}} \subseteq \mathcal{P}$, by the monotonicity of the greatest fixed point operator. Since $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$ we have $\mathcal{P} = \Phi^{\mathrm{S}}(\mathrm{S}(\mathcal{P})) \subseteq \Phi^{\mathrm{S}}(\mathcal{P})$, by the monotonicity of Φ^{S}. Hence $\mathcal{P} \subseteq \nu\, \Phi^{\mathrm{S}}$, by coinduction.

Case $(\mu\, \Phi)(\vec{t})$ *where* Φ *is not guarded.* $\mathrm{S}(\mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S})) \subseteq \mu\, (\Phi^{\mathrm{S}} \circ \mathrm{S})$ is shown in a similar way as $\mathrm{S}(\mathcal{P}) \subseteq \mathcal{P}$ was shown in the previous case, since there we used only the fixed point property.

Case $(\mu\, \Phi)(\vec{t})$ *where* Φ *is guarded.* We reason in CFP assuming $\Gamma_{\mu\, \Phi}$. Let Φ be $\lambda X \lambda \vec{x}\,.\, A$. Since X is guarded in A, we have $\Gamma_{\mu\, \Phi} = \Gamma_A$. Hence, by induction hypothesis, $\mathrm{S}(A^{\mathrm{S}}) \to A^{\mathrm{S}}$, i.e. $\mathrm{S}(\Phi^{\mathrm{S}}(X)) \subseteq \Phi^{\mathrm{S}}(X)$. Consequently, $\mathrm{S}(\mu\, \Phi^{\mathrm{S}}) = \mathrm{S}(\Phi^{\mathrm{S}}(\mu\, \Phi^{\mathrm{S}})) \subseteq \Phi^{\mathrm{S}}(\mu\, \Phi^{\mathrm{S}}) = \mu\, \Phi^{\mathrm{S}}$. ◀

Theorem 14 (Concurrent embedding). If $\Gamma \vdash_{\mathrm{IFP}} A$, then $\Gamma^{\mathrm{S}} \vdash_{\mathrm{CFP}} A^{\mathrm{S}}$ for all formulas A without free predicate variables.

Proof. Induction on derivations.

The assumption rule is trivial and the rules for the connectives where the embedding is defined homomorphically, that is, conjunction, implication, universal quantification as well as induction and coinduction, are straightforward using the induction hypothesis.

Disjunction introduction.

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash A^{\mathrm{S}}$. Hence $\Gamma^{\mathrm{S}} \vdash A^{\mathrm{S}} \vee B^{\mathrm{S}}$ by disjunction introduction. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}})$ by the rule (S^+)

Disjunction elimination.

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(A^{\mathrm{S}} \vee B^{\mathrm{S}})$, $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash C^{\mathrm{S}}$, and $\Gamma^{\mathrm{S}}, B^{\mathrm{S}} \vdash C^{\mathrm{S}}$. By the last two sequents, the rule (S^+) and disjunction elimination, we have $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vee B^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$, by the rule (S^-). With Lemma 13 it follows $\Gamma^{\mathrm{S}} \vdash C^{\mathrm{S}}$.

Existence introduction.

$$\frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x\, A}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash (A[t/x])^{\mathrm{S}}$. By existence introduction and since $(A[t/x])^{\mathrm{S}}$ is the same as $A^{\mathrm{S}}[t/x]$ it follows $\Gamma^{\mathrm{S}} \vdash \exists x\, A^{\mathrm{S}}$. Hence $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(\exists x\, A^{\mathrm{S}})$, by the rule (S^+).

Existence elimination.

$$\frac{\Gamma \vdash \exists x\, A \qquad \Gamma, A \vdash C}{\Gamma \vdash C}$$

By induction hypothesis, $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(\exists x\, A^{\mathrm{S}})$ and $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash C^{\mathrm{S}}$. Applying the rule (S^+) to the second sequent yields $\Gamma^{\mathrm{S}}, A^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$ and furthermore $\Gamma^{\mathrm{S}}, \exists x\, A^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$, using existence elimination (since the embedding and the modality don't introduce new free variables). With the rule (S^-) it follows $\Gamma^{\mathrm{S}} \vdash \mathrm{S}(C^{\mathrm{S}})$ and hence $\Gamma^{\mathrm{S}} \vdash C^{\mathrm{S}}$, using Lemma 13. ◀

Theorem 20 (Faithfulness). If $a \in D$ concurrently realizes a data formula A, then $\mathrm{data}(a)$ is nonempty and all $d \in \mathrm{data}(a)$ realize A, provably in IFP.

Proof. For an $n+1$-ary predicate P whose first argument is of type D we define a predicate P' of the same arity by $P'(a, \vec{x}) := \forall d \in \mathrm{data}(a)\, P(d, \vec{x})$, and extend this to predicate substitutions by setting $\theta'(\tilde{X}) := (\theta(\tilde{X}))'$. We show that for a parametric data formula A

$$\mathbf{r}(A^{\mathrm{S}})\theta' \subseteq (\mathbf{r}(A)\theta)'$$

In particular, for a data formula A we have $\mathbf{r}(A^{\mathrm{S}}) \subseteq \mathbf{r}(A)'$, i.e. if a realizes A^{S}, then $\mathbf{r}(A)'(a)$ holds, that is, d realizes A for all $d \in \mathrm{data}(a)$. The proof is by induction on A. We only look at the case $A = \mu\, \Phi\, \vec{t}$, since the other cases are easy. We show by induction $\mu\, (\mathbf{r}(\Phi^{\mathrm{S}})\, \theta') \subseteq P'$ where $P := \mu\, (\mathbf{r}(\Phi)\, \theta)$. Hence we have to show $(\mathbf{r}(\Phi^{\mathrm{S}})\, \theta')\, P' \subseteq P'$:

$$(\mathbf{r}(\Phi^{\mathrm{S}})\, \theta')\, P' = (\mathbf{r}(\Phi^{\mathrm{S}})\, \tilde{X})\, (\theta'[\tilde{X} := P])' \overset{\text{i.h.}}{\subseteq} (\mathbf{r}(\Phi)\, \tilde{X})\, (\theta'[\tilde{X} := P])' = (\mathbf{r}(\Phi)\, \theta)\, P)' = P'\,. \blacktriangleleft$$

Lemma 22 (Correctness).
(a) If $c \longrightarrow v$, then $\vdash \overline{c} = \overline{v}$.
(b) If $c \Longrightarrow d$, then $\vdash d \in \mathrm{data}([\![\overline{c}]\!])$.

Proof. Easy, by induction along the definitions of $c \longrightarrow v$ and $c \Longrightarrow d$. ◀

Lemma 23 (Instantiation).
(a) If $c[i/\alpha] \Longrightarrow d$, then $c \Longrightarrow d$.
(b) If $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \Longrightarrow d$, then $c \Longrightarrow d$.

Proof. Part (a) is proved by induction on i. Part (b) follows immediately from (a). ◀

Lemma 24 (Irreducibility of values). For values v, v'

$$v \longrightarrow v' \quad \text{iff} \quad v = v' .$$

Proof. This follows immediately from the rules of the big-step reduction relation. ◀

Lemma 25 (Monotonicity). If a, b are compact elements in D such that $a \sqsubseteq b$, then $\mathrm{Cl}(a) \supseteq \mathrm{Cl}(b)$.

Proof. Induction on the maximum of $\mathbf{rk}(a)$ and $\mathbf{rk}(b)$. In the case $a = \mathrm{Fam}\,\varphi \sqsubseteq \mathrm{Fam}\,\psi$ with $\varphi\, i \sqsubseteq \psi\, i$ for all $i \in \mathcal{I}$, we have $\mathrm{Cl}(\varphi\, i) \supseteq \mathrm{Cl}(\psi\, i)$, by induction hypothesis. Let $c \in \mathrm{Cl}(b)$. We show $c \in \mathrm{Cl}(a)$. Let $c \longrightarrow (\langle\alpha\rangle M, \eta)$ such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\psi\, i)$ for all $i \in \mathcal{I}$. Hence $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$, which proves that $c \in \mathrm{Cl}(a)$.

The other cases are as in [1], Lemma 12. ◀

Lemma 26 (Printing of data). If $c \in \mathrm{Cl}(a)$ and $d \in \mathrm{data}(a)$, then $c \Longrightarrow d$.

Proof. Induction on the definition of $\mathrm{data}(a)$. In the case $a = \mathrm{Fam}\,\varphi$, the hypotheses of the lemma imply that we have $d \in \mathrm{data}(\varphi\, i_0)$ for some $i_0 \in \mathcal{I}$ and $c \longrightarrow (\langle\alpha\rangle M, \eta)$ such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Therefore, by induction hypothesis, $(M[i_0/\alpha], \eta) \Longrightarrow d$. By Lemma 23, $c \Longrightarrow d$.

The other cases are easy. ◀

Lemma 27 (Reducibilty of closures). $c \in \mathrm{Cl}(a)$ iff $c \longrightarrow v$ for some $v \in \mathrm{Cl}(a)$.

Proof. Induction on $\mathbf{rk}(a)$. We only consider the case $a = \mathrm{Fam}\,\varphi$ since the other cases are as in [1], Lemma 13.

If $c \in \mathrm{Cl}(a)$, then $c \longrightarrow (\langle\alpha\rangle M, \eta)$ and $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Set $v := (\langle\alpha\rangle M, \eta)$, which is a value. By rule (i), $v \longrightarrow v$. Hence $v \in \mathrm{Cl}(a)$, by the above.

Conversely, if $v \in \mathrm{Cl}(a)$, then $v \longrightarrow (\langle\alpha\rangle M, \eta)$ for some α, M, η such that $(M[i/\alpha], \eta) \in \mathrm{Cl}(\varphi\, i)$ for all $i \in \mathcal{I}$. Since $(\langle\alpha\rangle M, \eta)$ is a value, it follows, by Lemma 24, that $v = (\langle\alpha\rangle M, \eta)$. Hence, if $c \longrightarrow v$, then $c \in \mathrm{Cl}(a)$. ◀

Lemma 28 (Approximation). If $\eta \in \mathrm{Cl}(\xi)$, $a \in D_0$ and $a \sqsubseteq [\![M]\!]\xi$, then $(M,\eta) \in \mathrm{Cl}(a)$.

Proof. As in [1], Lemma 15, we replace in the statement of the lemma the value $[\![M]\!]\xi$ by its nth approximation $[\![M]\!]^n\xi$ and show,

$$(+) \qquad \text{for all } n \in \mathbb{N},\ \text{if } \eta \in \mathrm{Cl}(\xi), a \in D_0 \text{ and } a \sqsubseteq [\![M]\!]^n\xi,\ \text{then } (M,\eta) \in \mathrm{Cl}(a),$$

by induction on n. Since $([\![M]\!]^n\xi)_{n\in\mathbb{N}}$ is an increasing sequence with $[\![M]\!]\xi$ as its supremum, it follows that for compact a, $(+)$ is equivalent to the statement of the lemma.

We assume $a \neq \bot$ (for $a = \bot$ the statement is trivial) and look only at the cases where M is formed by one of the new constructs. Since $[\![M]\!]^0\xi = \bot$ we are in the step of the induction. Hence we assume $a \sqsubseteq [\![M]\!]^{n+1}\xi$.

Case $M = \langle\alpha\rangle N$. Then $[\![M]\!]^{n+1}\xi = \mathrm{Fam}\,\varphi$ where $\varphi\, i = [\![N[i/\alpha]]\!]^n\xi$. Since $a \neq \bot$ we have $a = \mathrm{Fam}\,\varphi_0$ where $\varphi_0\, i \sqsubseteq \varphi\, i$ and $\varphi_0\, i$ is compact for all $i \in \mathcal{I}$. Since $v := (M,\eta)$ is a value, we have $v \longrightarrow v$, and, by induction hypothesis, $(N[i/\alpha],\eta) \in \mathrm{Cl}(\varphi_0\, i)$. Hence $(M,\eta) \in \mathrm{Cl}(a)$.

Case $M = \mathrm{case}\ k\ \mathrm{of}\ \{* \to N_*\,;\, \mathrm{L}(\alpha) \to N_0\,;\, \mathrm{R}(\alpha) \to N_1\,;\, (\alpha,\beta) \to N\}$.

Subcase $k = *$. Then $[\![M]\!]^{n+1}\xi = [\![N_*]\!]^n\xi$. By induction hypothesis $(N_*,\eta) \in \mathrm{Cl}(a)$. By Lemma 27, $(N_*,\eta) \longrightarrow v$ for some value $v \in \mathrm{Cl}(a)$. Hence $(M,\eta) \longrightarrow v$.

Subcase $k = \mathrm{L}(i)$ for some $i \in \mathcal{I}$. Then $[\![M]\!]^{n+1}\xi = [\![N_p]\!]^n\xi[\alpha \mapsto i]$. By induction hypothesis $(N_0,\eta[\alpha \mapsto i]) \in \mathrm{Cl}(a)$. By Lemma 27 (implication from left to right), $(N_0,\eta[\alpha \mapsto i]) \longrightarrow v$ for some value $v \in \mathrm{Cl}(a)$. Hence $(M,\eta) \longrightarrow v$, By Lemma 27 (implication from right to left).

Subcase $k = \mathrm{R}(i)$ for some $i \in \mathcal{I}$. Similar to the case above.

Subcase $k = (i,j)$ for some $i,j \in \mathcal{I}$. Similar to the previous case.

Case $M = N\, i_0$. By assumption, $a \sqsubseteq [\![M]\!]^{n+1}\xi = [\![N]\!]^n\xi \cdot i_0$. Since $a \neq \bot$, $[\![N]\!]^n\xi = \mathrm{Fam}\,\varphi$, for some $\varphi \in D^{\mathcal{I}}$. Define $\varphi_0 \in D^{\mathcal{I}}$ by $\varphi_0(i_0) := a$ and $\varphi_0(i) := \bot$ for $i \neq i_0$. Then $\mathrm{Fam}\,\varphi_0$ is compact and $\mathrm{Fam}\,\varphi_0 \sqsubseteq [\![N]\!]^n\xi$. By induction hypothesis, $(N,\eta) \in \mathrm{Cl}(\mathrm{Fam}\,\varphi_0)$. Hence $(N,\eta) \longrightarrow (\langle\alpha\rangle N_0,\eta) \in \mathrm{Cl}(\varphi_0\, i_0) = \mathrm{Cl}(a)$. By Lemma 27, $N_0[i_0/\alpha] \longrightarrow v \in \mathrm{Cl}(a)$ and consequently $N\, i_0 \longrightarrow v$. Therefore $N\, i_0 \in \mathrm{Cl}(a)$. ◀

Lemma 29. If $d \in \mathrm{data}(a)$, then $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq a$.

Proof. Easy induction on the definition of $d \in \mathrm{data}(a)$. We only look at rule (iv), that is, $d \in \mathrm{Fam}\,\varphi$ because $d \in \mathrm{data}(\varphi\, i_0)$ for some $i_0 \in \mathcal{I}$. By induction hypothesis, $d \in \mathrm{data}(a_0)$ for some compact $a_0 \sqsubseteq \varphi\, i_0$. Define $\varphi_0 \in D^{\mathcal{I}}$ by $\varphi_0\, i_0 := a$ and $\varphi_0\, i := \bot$ for $i \neq i_0$. Then $\mathrm{Fam}\,\varphi_0$ is compact and $\mathrm{Fam}\,\varphi_0 \sqsubseteq \mathrm{Fam}\,\varphi$, and, by rule (iv), $d \in \mathrm{data}(\mathrm{Fam}\,\varphi_0)$. ◀

Polymorphic Game Semantics for Dynamic Binding

James Laird*

Department of Computer Science, University of Bath, U.K.

Abstract

We present a game semantics for an expressive typing system for block-structured programs with late binding of variables and System F style polymorphism. As well as generic programs and abstract datatypes, this combination may be used to represent behaviour such as dynamic dispatch and method overriding.

We give a denotational models for a hierarchy of programming languages based on our typing system, including variants of PCF and Idealized Algol. These are obtained by extending polymorphic game semantics to block-structured programs. We show that the categorical structure of our models can be used to give a new interpretation of dynamic binding, and establish definability properties by imposing constraints which are identical or similar to those used to characterize definability in PCF (innocence, well-bracketing, determinacy). Moreover, relaxing these can similarly allow the interpretation of side-effects (state, control, non-determinism) - we show that in particular we may obtain a fully abstract semantics of polymorphic Idealized Algol with dynamic binding by following exactly the methodology employed in the simply-typed case.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases Game semantics, denotational models, PCF, Idealized Algol

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.27

1 Introduction

We describe a semantics for higher-order programs with combinations of three features: System F-style polymorphism, block structured *late binding*, and side-effects (control and state). Generic, higher-rank polymorphism is a powerful principle supporting information hiding and encapsulation, through features such as generics and interfaces, modules or abstract data types. It is frequently encountered in combination with various side-effects, and in contexts in which binding of variables to values (and, implicitly or explicitly to types) is late, or dynamic, – i.e. dependent on where they are called. Examples include features such as dynamic dispatch, virtual methods and method overriding.

Previous denotational models of parametric polymorphism have focussed on λ-calculus-based (and effect-free) type theories with static binding such as System F [7, 23]. Late binding has received less theoretical attention, and there is a lack of formal principles on which to base construction of a model. The aims of this work are to develop such principles for an *intensional semantics* of generic polymorphism, to show that they can be used to capture program behaviour in the presence or absence of side-effects such as local state and non-local control, and to characterise program equivalence precisely and concretely, as a step towards semantics-based verification.

* Research supported by EPSRC grant EP/K037633/1.

© James Laird;
licensed under Creative Commons License CC-BY

Table 1 Typing Judgements.

$$\frac{\Theta \vdash \Gamma, T}{\Gamma, x{:}T, \Gamma' \vdash_\Theta x{:}T} \qquad \frac{\Gamma, x{:}S, \Gamma' \vdash_\Theta N{:}T \qquad \Gamma, x{:}S, \Gamma' \vdash_\Theta M{:}S}{\Gamma, x{:}S, \Gamma' \vdash_\Theta \mathtt{let}\ x{=}M\ \mathtt{in}\ N{:}T} \qquad \frac{\Gamma \vdash_{\Theta, X} M{:}T \qquad \Theta \vdash \Gamma}{\Gamma \vdash_\Theta \Lambda X.M{:}\forall X.T}$$

$$\frac{\Gamma, x{:}S \vdash_\Theta M{:}T}{\Gamma \vdash_\Theta \lambda x^S.M{:}S{\to}T} \qquad \frac{\Gamma \vdash_\Theta M{:}S{\to}T \qquad \Gamma \vdash_\Theta N{:}S}{\Gamma \vdash_\Theta M\ N{:}T} \qquad \frac{\Gamma \vdash_\Theta M{:}\forall X.T \qquad \Theta \vdash S}{\Gamma \vdash_\Theta M\{S\}{:}T[S/X]}$$

Our model extends the game semantics for generic call-by-name polymorphism described in [17], which developed earlier notions of variable game [10, 2] with an interpretation of quantification as a relation between question and answer moves. It was used in [17] to construct a fully abstract semantics of a programming language with a typing system based on System F, and locally declared *general references*. The capacity to escape from static binding using references is fundamental to the full abstraction result, posing the question of whether there are more constrained models of polymorphism based on similar principles. We address this question by giving a model of a language based on PCF and its game semantics [12, 4], with dynamic binding (as in dialects of Lisp, for example). This has a block structure – i.e. definitions are scoped by the block in which they occur, which is captured by our games model in a natural way, whereas instantiation breaks *static* scope, suggesting a semantic relationship between dynamic scope and polymorphism. We develop a formal semantics of late binding using the categorical properties of our model – in particular, it is an instance of *sequoid structure*, which was introduced in [15].

We also consider the extension of our language with side-effects. In a close analogy with the games models of PCF with static binding, we identify three conditions on strategies (determinacy, first-order well-bracketing and dynamic innocence) required for our definability result, which may be relaxed individually, or in combination, to interpret side-effects (non-determinism, non-local control flow, integer state). We focus on the last of these, giving a fully abstract semantics of Idealized Algol [24] – a block-structured language with integer state, now extended with System F-style polymorphism and dynamic binding, a combination which can be used to represent objects and classes [22]. Our model gives a very concrete representation of polymorphic types, suggesting that it could lend itself to algorithmic models for determining program equivalence and other properties, as in e.g. [6].

2 A Polymorphic Type Theory with Dynamic Binding

We define a second-order type theory with late binding by extending System F [23, 7] with a set of ground types B and a (*non-binding*) `let` operation. The formation rules for types are:

$$\frac{}{\Theta, X, \Theta' \vdash X} \qquad \frac{}{\Theta \vdash B} \qquad \frac{\Theta \vdash S \qquad \Theta \vdash T}{\Theta \vdash S \to T} \qquad \frac{\Theta, X \vdash T}{\Theta \vdash \forall X.T}$$

Typing rules for terms are given in Table 1.

Universally quantified types may be used to type generic programs [23], but also to construct new datatypes [7] including abstract data types, modules and interfaces, represented using existential types as described in [20] – e.g. the type $\exists X.(T_1 \& \ldots \& T_n) =_{df} \forall Y.(\forall X.T_1 \to \ldots \to T_n \to Y) \to Y$ may be assigned to objects which implement methods of type $T_1, \ldots, T_n$ which interface with a shared, abstract type X.

We define a programming language, PCF^2_d (second-order PCF with dynamic binding) based on this type theory by adding appropriate constants for arithmetic: $0 : \mathtt{nat}$, $\mathtt{succ}, \mathtt{pred} : \mathtt{nat} \to \mathtt{nat}$, divergence $(\Omega_\mathtt{T} : T)$ and conditionals $\mathtt{If0} : \mathtt{nat} \to T \to T \to T$. This contains

Table 2 Operational Semantics of PCF^2_d.

$$\frac{}{C;\mathcal{E}\Downarrow C;\mathcal{E}} \qquad \frac{M;\mathcal{E}\Downarrow 0;\mathcal{E}'}{\mathtt{If0}\ M;\mathcal{E}\Downarrow\lambda xy.x;\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow\lambda x.M';\mathcal{E}'_a \quad M'[a/x];\mathcal{E}'_a,(a,N)\Downarrow C;\mathcal{E}''}{M\ N;\mathcal{E}\Downarrow C,\mathcal{E}''} \qquad \frac{M;\mathcal{E},(a,M),\mathcal{E}_a\Downarrow C;\mathcal{E}''}{a;\mathcal{E},(a,M),\mathcal{E}_a\Downarrow C;\mathcal{E}''}$$

$$\frac{M;\mathcal{E}\Downarrow \mathtt{n+1};\mathcal{S}';\mathcal{E}'}{\mathtt{pred}\ M;\mathcal{E}\Downarrow \mathtt{n};\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow \mathtt{n+1};\mathcal{E}'}{\mathtt{If0}\ M;\mathcal{E}\Downarrow\lambda xy.y;\mathcal{E}'} \qquad \frac{M;\mathcal{E}\Downarrow\Lambda X.M';\mathcal{E}' \quad M'[T/X];\mathcal{E}'\Downarrow C;\mathcal{E}''}{M\{T\};\mathcal{E}\Downarrow C,\mathcal{E}''} \qquad \frac{N;\mathcal{E},(a,M)\Downarrow C;\mathcal{E}'}{\mathtt{let}\ a{=}M\ \mathtt{in}\ N;\mathcal{E}\Downarrow C;\mathcal{E}'}$$

PCF itself: note that `let` definitions are implicitly recursive – thus we may define fixed points (e.g. $\mathbf{Y} : (T \to T) \to T =_{df} \lambda f^{T\to T}.(\lambda x^T.\mathtt{let}\ x = (f\ x)\ \mathtt{in}\ x)\ \Omega$).

The operational semantics for PCF^2_d (Table 2) is given by a convergence relation between configurations $M;\mathcal{E}$ and $C;\mathcal{E}'$ where:

- M and C are terms of PCF^2_d extended with a set of *procedure names*.
- C is in *canonical form*, given by the grammar: $C ::= \mathtt{n} \mid \lambda x.M \mid \Lambda X.M$
- $\mathcal{E}, \mathcal{E}'$ are finite *sequences* $(a_1, M_1), (a_2, M_2), \ldots, (a_n, M_n)$ of bindings of procedure names to terms (which may contain more than one binding for each name). $\mathcal{E}_a$ denotes a stack which does not contain a binding for a. We write $M \Downarrow \mathtt{n}$ if $M;_ \Downarrow \mathtt{n};\mathcal{E}$ for some $\mathcal{E}$.

Here are some examples clarifying the expressive power of our language:

- Late binding can be used to vary the meaning of a procedure, depending on the (dynamic) scope in which it is called – for example, in $\mathtt{let}\ f = (\mathtt{let}\ x = 1\ \mathtt{in}\ M)\ \mathtt{in}\ \mathtt{let}\ x = 0\ \mathtt{in}\ N$, the variable x takes the value 1 if called inside the scope of f, and 0 otherwise. Note that this is an instance of *overriding* of one definition of x by another.
- Like general references, late binding can allow the arguments of a function to escape from its static scope – for example $g : (R \to S \to S) \to R \to T) \vdash \lambda x^R.(g\ (\lambda y^T.\lambda z^S.\mathtt{let}\ x = y\ \mathtt{in}\ z))\ x : R \to T$ – in which the second argument to g may be captured from inside the scope of the first.
- Unlike references, late binding does not allow procedures to pass information out of the block in which they are defined. So, for example the functions $\lambda x^{\mathtt{nat}}.(\mathtt{If0}\ x)\ \mathtt{then}\ x\ \mathtt{else}\ x$ and $\lambda x^{\mathtt{nat}}.x$ are observationally equivalent in PCF^2_d (as in PCF itself), as no information can be passed between sequential invocations of x. As in e.g. Idealized Algol adding integer state (see Section 7) allows these terms to be separated.
- Late binding of terms extends implicitly to types – e.g. a variable of type $\exists X.T$ may become bound to procedures in which the explicit witnessing type varies depending on where the binding takes place.

3 Simple Games

Our model is based on a games interpretation of second-order (intuitionistic) linear type theory described in [17], adapted to modelling block structure and dynamic scope. It may also be constructed in a setting with explicit justification pointers, closer to the original model of PCF [12] and models based on relaxing the constraints on its strategies [4, 14, 9] – comparison with the latter would be one reason for preferring such a construction. Our interpretation of second-order types adapts readily to such a setting [16], where we may give a combinatorial definition of the interpretation of dynamic binding along the lines of the semantics of general references in [5]. However, the presentation here via a model of intuitionistic linear type theory allows a categorical decomposition of our model using structures introduced in [15], which are useful in proving its soundness.

We begin by describing our semantics of dynamic binding at first-order types, which may be interpreted as *simple games*, given by a set of moves partitioned between two players, and

between questions and answers, together with event-structure-style *enabling* and *conflict* relations on moves. We work with a *universal* set of moves, for which these relations and predicates are fixed, principally because this will ultimately simpify the definition of type substitution.

Fix a set of *tags* $\mathcal{A}$ which includes left and right tags l, r and special "enabling" and "conflict" tags e, c), and a set of *values* $\mathcal{V}$ which includes the natural numbers. The set $\mathcal{U}$ of *moves* is the set of sequences $\mathcal{A}^* \cup \{w!_v \mid w \in \mathcal{A}^*, v \in \mathcal{V}\}$. For any set of moves X, we write X^+ and X^- f for the subsets of X consisting of Opponent and Proponent moves (even and odd-length sequences, respectively) and $X^?$ and $X^!$ for the sets $X \cap \mathcal{A}^*$ of (*first-order questions* or 1-questions) and $X \backslash \mathcal{A}^*$ (*first-order answers* or 1-answers). These play the same roles as questions and answers in the game semantics of PCF [3, 12] and Idealized Algol [4] – opening and closing blocks of moves, corresponding to procedure calls. We will subsequently introduce a further Q/A-labelling to capture second order structure, effectively giving two levels of bracketing on games.

▶ **Definition 1.** A *simple game* is a set of moves $A \subseteq \mathcal{U}$ such that if $u \cdot w \in A$ then $u \in A$ if and only if $w = \varepsilon$ or $w = !_v$ for some $v \in \mathcal{V}$.

In other words, for every 1-answer $m!_v \in A^!$, there is a corresponding 1-question $m \in A^?$ (belonging to the other player), but moves are otherwise incomparable with respect to the prefix order. For example, given a set of values $X \subseteq \mathcal{V}$, $\underline{X} = \{\varepsilon\} \cup \{!_v \mid v \in X\}$ is the "flat" game with a single, Opponent 1-question ε, and a corresponding Proponent 1-answer $!_v$ for each $v \in X$ (corresponding to an atomic datatype with values in X). We define the following relations on 1-questions (cf. *event structures*, although conflict is reflexive):

Enabling: $m \vdash n$ if there exist $u \in \mathcal{A}^*$, $w, w' \in (\mathcal{A}\backslash\{e\})^*$ such that $m = u \cdot w$ and $n = u \cdot ew'$.

Conflict: $m \# n$ if $m = n$ or there exist $u, w, w' \in \mathcal{A}^*$ such that $m = u \cdot cw$ and $n = u \cdot cw'$.

A move $m \in (\mathcal{A}\backslash\{e\})^*$ is *initial*. A game A is *negative* if every initial move in M is an Opponent move. We construct simple games using the following operations:

- For a game A, and sequence $w \in \mathcal{A}^*$, $w.A = \{w \cdot m \mid m \in A\}$ is a game.
- For games A, B such that $m \not\sqsubseteq m'$ and $m \not\sqsubseteq m'$ for all $(m, m') \in A \times B$, $A \cup B$ is a game.
- $A \& B = cl.A \cup cr.B$ – a disjoint union of A and B with initial moves in conflict.
- $A \otimes B = ll.A \cup rr.B$ – a disjoint union of A and B with initial moves not in conflict.
- $A \to B = l.A \cup rr.B$ a disjoint union of A and B with initial moves not in conflict and Proponent/Opponent roles switched in A.

Positions of a game are represented as finite sequences of moves subject to certain conditions on the *stack of open* 1*-questions* – the subsequence obtained by erasing all moves between any answer and its corresponding question (inclusive) – i.e. $\mathsf{stack}(sq) = \mathsf{stack}(s)q$ if $q \in \mathcal{U}^?$, and $\mathsf{stack}(s(q!_v)) = s'$ if $\mathsf{stack}(s) = s'qt$ and q doesn't occur in t. ($\mathsf{stack}(s)$ is undefined, otherwise). In other words playing a 1-question move pushes it onto the stack, corresponding to opening a block, and playing an answer pops the corresponding question from the stack, closing the block. Valid positions of the game (*legal sequences*) are defined by requiring that in the stack every move is preceded by an enabling move (i.e. every call to an argument is preceded by an open call to its originating function) and no open moves are in conflict.

▶ **Definition 2.** The set L_A of legal sequences on A consists of $t \in A^*$ such that if $sn \sqsubseteq t$:

Alternation: n is an Opponent move if and only if s is even length.

Conflict-freeness: If $s'm \sqsubseteq \mathsf{stack}(s)$ then $m \not\# n$

Enabling: If m is a non-initial 1-question then $m \vdash n$ for some $s'm \sqsubseteq \mathsf{stack}(s)$.

A *strategy* on A is a non-empty, even-prefix-closed set of even-length sequences in L_A.

The strategies in our model satisfy a further constraint – they depend only on the moves in the current (open) block.

▶ **Definition 3.** A sequence $s \in L_A$ is *complete* if $\mathsf{stack}(s) = \varepsilon$. A strategy σ is $*$*-closed* if whenever $s \in \sigma$ is complete then for any $t \in L_A$, $s \cdot t \in \sigma$ if and only $t \in \sigma$.[1]

We now define a series of additional constraints on strategies which will combine to yield definability in PCF^2_d. These are analogous (or identical) to the constraints used to characterize definability in PCF itself [12]. First, we define a restricted history of play accessible to strategies definable with dynamic binding – a version of the *view function* (which may be seen as a representation of *static* scope) which considers only the (implicit) justification pointers between 1-answers and their corresponding questions.

▶ **Definition 4.** The (Proponent) *dynamic view* $\lceil s \rceil$ of $s \in L_A$ is defined as follows:
$\lceil sq \rceil = \lceil s \rceil q$, if $q \in A^?$ (q is a question),
$\lceil s(q!_v) \rceil = s'q(q!_v)$, if $(q!_v) \in A^+$ ($q!_v$ is an Opponent 1-answer) and $\lceil s \rceil = s'qt$,
$\lceil s(q!_v) \rceil = s'$, if $(q!_v) \in A^-$ ($q!_v$ is a Proponent 1-answer) and $\lceil s \rceil = s'qt$.

▶ **Definition 5.** A $*$-closed strategy σ on a simple game A is:

- *deterministic* if $sa, sb \in \sigma$ implies $a = b$.
- *dynamically innocent* if it is deterministic and $sab, tab \in L_A$, $sab, t \in \sigma$ and $\lceil sa \rceil = \lceil ta \rceil$ implies $tab \in \sigma$. In other words, play by σ may only depend on the dynamic view.[2]
- 1*-well-bracketed* if $s(m!_v) \in \sigma$ implies $\mathsf{stack}(s) = s'm$ – in other words, Proponent may only answer the most recently asked, unanswered question (close the most recently opened enclosing block).

We abbreviate combinations of these constraints as combinations of the letters $\{B, D, I\}$.

So, for example, the only deterministic $*$-closed strategies on the game $\underline{\mathbb{N}}$ are the dynamically innocent and well-bracketed strategies $\bot = \{\varepsilon\}$ and for each $i \in \mathbb{N}$ the strategy $(\varepsilon \cdot !_i)^*$ which responds to Opponent's initial 1-question ε with the answer $!_i$ each time it is asked.

3.1 Categories of Simple Games

For each combination C of constraints (B, D, I) we construct a category $\mathcal{G}_C$, of (negative) simple games, in which morphisms from A to B are $*$-closed C-strategies on $A \to B$.

▶ **Definition 6.** Given a sequence s over A, and an evident partial projection function from A to B, we write $s{\restriction}B$ for the sequence obtained by applying the projection to the moves on which it is defined, and omitting moves on which it is not defined. Given sets of moves $X, Y \subseteq \mathcal{U}$, a sequence s is a *Proponent* X,Y-copycat if for any *even-length* prefix $t \sqsubseteq s$, $t{\restriction}X = t{\restriction}Y$ and an *Opponent* X,Y-copycat if for any *odd-length* prefix $t \sqsubseteq s$, $t{\restriction}X = t{\restriction}Y$.

The identity $\mathsf{id}_A : A \to A$ consists of legal sequences which are Proponent (A^+, A^-) copycats.

▶ **Definition 7.** An *interaction sequence* on games A_1, A_2, A_3 is a sequence s on $A_1 \uplus A_2 \uplus A_3$ such that $s \restriction A_i \to A_j$ is legal for each $i < j$. The *composition* of $\sigma : A_1 \to A_2$ with $\tau : A_2 \to A_3$ is the set of legal sequences t on $A_1 \to A_3$ such that there exists an interaction sequence s on A_1, A_2, A_3 with $s{\restriction}A_1 \to A_2 \in \sigma$ and $s{\restriction}A_2 \to A_3 \in \tau$, and $t = s{\restriction}A_1 \to A_3$.

[1] Cf. the notion of single-threadedness [9].
[2] Note that dynamic innocence implies determinacy by definition: naive attempts to form a category of nondeterministic dynamic-innocent strategies fail for precisely the reasons described in [8] for nondeterministic innocent strategies.

This is a simplified but equivalent definition of composition to that given in [17] (we use the latter, in combination with the following property of the stack, to prove associativity):

▶ **Lemma 8.** *If s is an interaction sequence then* $\mathsf{stack}(s\restriction A_i \to A_j) = \mathsf{stack}(s)\restriction A_i \to A_j$.

Using the proofs of *compositionality* of bracketing, determinacy, and innocence (of which dynamic innocence is a variant) [18, 14, 8], we show:

▶ **Proposition 9.** *For each constraint C, the composition of C-strategies is a C-strategy.*

Since the identity is a BDI-strategy, we have a category $\mathcal{G}_C$ for each constraint. We define symmetric monoidal structure on $\mathcal{G}_C$: $A \otimes B$ is the disjoint union $ll.A \cup rr.B$ with unit $I = \varnothing$ (which is a terminal object). Given $\sigma : A \to C$ and $\tau : B \to D$, we define $\sigma \otimes \tau : A \otimes B \to C \otimes D = \{s \in L_{A\otimes B \to C \otimes D} \mid s\restriction A \to C \in \sigma \wedge s\restriction B \to D \in \tau\}$.

$\mathcal{G}_C$ is not symmetric monoidal closed but does have sufficient exponentials to allow us to construct a Cartesian closed category based on its cofree commutative comonoids. Say that a negative game is *well-opened* if any two of its initial moves are in conflict (so any legal sequence contains at most one unanswered initial question).

▶ **Proposition 10.** *The well-opened games form an* exponential ideal *in $\mathcal{G}_C$.*

Proof. Given a well-opened game B, the (well-opened) exponential of a game A by B is $A \multimap B = e.A \cup rr.B$ – a disjoint union of A and B in which Opponent and Proponent swap roles in A, and initial moves of A are enabled by initial moves of B. ◀

▶ **Proposition 11.** *Any family of well-opened games has a well-opened cartesian product in $\mathcal{G}_C$.*

Proof. E.g. if A_1, A_2 are well-opened then $A_1 \& A_2 = cl.A_1 \cup cr.A_2$ is a (well-opened) cartesian product of A_1 and A_2: by conflict-freeness, any legal sequence on $B \to A_1 \& A_2$ has the form $t \cdot s$, where t is a complete sequence and s is a sequence in $B \to A_1$ or $B \to A_2$. So by the $*$-closure condition, $\sigma : B \to A_1 \& A_2$ is uniquely determined by its left and right projections. ◀

The rules defining legal plays allow the repeated *sequential* calling of procedures, since a 1-question may be replayed once it has been closed, but not – for example – sharing between function and argument. To capture the latter, we show that well-opened games possess a *cofree commutative comonoid* in $\mathcal{G}_C$ based on the following construction:

▶ **Definition 12.** For a well-opened game A let $!A$ be the well-opened game $(er)^*.ll.A$.

In other words $!A = ll.A \cup erll.A \cup ererll.A \cup \ldots$ consists of countably many distinctly tagged copies of A: initial moves in each copy $(er)^n.ll.A$ must be played (by Opponent) in order of the size of n – i.e. it is an instance of a construction introduced in [11].

We may define copycat morphisms $\delta_A : !A \to !A \otimes !A$ and $\epsilon_A : !A \to I$ making $(!A, \delta_A, \epsilon_A)$ a *commutative comonoid*: informally, δ_A plays copycat between $!A \otimes !A$ and $!A$, opening a fresh copy of A in $!A$ for each thread opened on either side of $!A \otimes !A$ (ϵ_A is the strategy $\{\varepsilon\}$). In section 5 we will use the categorical properties of $\mathcal{G}_C$ to define these morphisms formally, and moreover to show that $(!A, \delta_A, \epsilon_A)$ is the *cofree commutative comonoid* on A in $\mathcal{G}_C$ [13] (Proposition 22). In other words, if $U : \mathsf{Comon}(\mathcal{G}_C) \to \mathcal{G}_C$ is the forgetful functor into $\mathcal{G}_C$ from its category of commutative comonoids, then for any object $(B, \delta_B, \epsilon_B)$ of $\mathsf{Comon}(\mathcal{G}_C)$ there is an equivalence (natural in B) between $\mathcal{G}_C(U(B), A)$ and $\mathsf{Comon}(\mathcal{G}_C)(B, !A)$ given by a morphism $\mathsf{der}_A : U(!A) \to A$ and an operation sending each morphism $f : U(B) \to A$

in $\mathcal{G}_C$ to $f^\dagger : B \to !A$ such that $f^\dagger; \mathsf{der}_A = f$ and $\mathsf{der}_A^\dagger = \mathsf{id}_{!A}$. Thus we have a comonad on the full subcategory $\mathcal{G}_C^W$ of $\mathcal{G}_C$ consisting of well-opened games, with co-Kelisli triple $(!_, \mathsf{der}, (_)^\dagger)$. By Lemmas 10 and 11 its co-Kleisli category is Cartesian closed.

4 Second-Order Games

Second-order games are based on the model in [17], which represents unbound type variables as $\mathbb{N}$-indexed "holes" into which games may be plugged, and quantification as a labelling of these holes as questions or answers. However, we need to extend this interpretation to take account of block structure (type variables are represented as generic blocks with an opening and closing 1-question). This "second-order" Q/A-labelling is imposed on top of the existing, intrinsic, question-answer structure of simple games (and the constraints of dynamic innocence and 1-well-bracketing):

▶ **Definition 13.** A *second-order game* A is a tuple $(|A|, \lambda_A, \rhd_A)$, where:

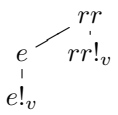
- $|A| \subseteq \mathcal{U}$ is a simple game.
- $\lambda_A : |A|^? \rightharpoonup (\{\mathsf{Q}, \mathsf{A}\} \cup \mathbb{N}\backslash\{0\}$ is a *partial* labelling of 1-questions as 2-*questions*, 2-*answers* or i-*holes* for $i \in \mathbb{N}\backslash\{0\}$
- $\rhd_A \subseteq |A|^? \times |A|^?$ is a *Q/A-relation* on 1-questions such that $\rhd_A \subseteq (\lambda_A^{-1}(\mathsf{Q})^- \times \lambda_A^{-1}(\mathsf{A})^+) \cup (\lambda_A^{-1}(\mathsf{Q})^+ \times \lambda_A^{-1}(\mathsf{A})^-)$ (i.e. Proponent answers Opponent questions and vice-versa).

2-questions and 2-answers represent bound variables which may be instantiated by the questioner. We say that A is a n-*context* game if the set $\lambda_A^{-1}(i)$ of i-hole moves (representing the ith free type variable) is empty for $i > n$ (it is *closed* if $n = 0$).

Note that some 1-questions may be unlabelled – these correspond to concrete ground types. In particular, every simple game A corresponds to a n-context game $(A, \varnothing, \varnothing)$ and the constructions on simple games extend to context games. In addition:

- For $1 \leq i \leq n$, let $\bullet_i$ be the n-context game $(\underline{\mathcal{V}}, \{(\varepsilon, i)\}, \varnothing)$ (corresponding to the type variable X_i), with a single (i-hole) 1-question, ε, with 1-answers $!_v$ for each $v \in \mathcal{V}$.
- $\forall_n A = (|A|, \lambda_A[\lambda_A^{-1}(n)^- \mapsto \mathsf{Q}, \lambda_A^{-1}(n)^+ \mapsto \mathsf{A}], \rhd_A \cup (\lambda_A^{-1}(n)^- \times \lambda_A^{-1}(n)^+)$ – the Proponent n-hole moves of A become answers to the Opponent n-hole moves, which become questions.

For example, the game $\Sigma = \forall_1(\bullet_1 \multimap \bullet_1)$ is played over the simple game $\{rr, e\} \cup \{rr!_v, e!_v \mid v \in \mathcal{V}\}$; rr is a 2-question, with 1-answers $rr!_v$ and 2-answer e, which has 1-answer $e!_v$ for $v \in \mathcal{V}$:



Contrast with the corresponding game in [17] (which has a single question and answer).

▶ **Definition 14.** The set L_A of legal sequences on A consists of $t \in L_{|A|}$ such that:

Well-bracketing (2): If $\lambda_A(m) = A$ then $\mathsf{pending}(\mathsf{stack}(s)) = s'q$, where $q \rhd_A n$ and $\mathsf{pending}(s)$ is the last-asked, unanswered 2-question in s.[3]

Q/A-copycat: If $sm(q!_v) \sqsubseteq t$ (where $\lambda_A(q) = \mathsf{Q}$) or $s(q!_v)m \sqsubseteq t$ (where $\lambda_A(q) = \mathsf{A}$) then $m = q'!_v$ for some q'.

A second-order strategy is a non-empty even-prefix-closed set of even-length legal sequences which satisfies the *Q/A-copycat condition for strategies*: if $s \in \sigma$ and $s(q!_v) \in L_A$, where $\lambda_A(q) = \mathsf{A}$ then $s(q!_v)(q'!_v) \in \sigma$ for some q'.

[3] i.e. $\mathsf{pending}(sm) = \mathsf{pending}(s')$ if $\lambda_A(m) = \mathsf{A}$ and $\mathsf{pending}(s) = s'q$; $\mathsf{pending}(sm) = sm$ otherwise.

In other words, the stack obeys a well-bracketing condition for 2-questions and answers as defined in [17] – calls to procedures of generic type are matched as opening and closing parentheses – and the new Q/A-copycat condition. This requires that the 1-answer move which closes a 2-question must copy the preceding 1-answer (so, for example, Proponent cannot immediately close Opponent's opening question in Σ with a 1-answer) and that a 1-answer which closes a 2-answer must be copied by the following move. In combination, they force the propagation of return values from 2-answers up to the corresponding 2-question. For an example of how the well-bracketing and Q/A-copycat conditions bite, observe that the only strategies on $\Sigma = \forall_1(\bullet_1 \multimap \bullet_1)$ are $\bot = \{\varepsilon\}$ and the "generic identity"copycat strategy $\top : I \to \Sigma$ consisting of even prefixes of sequences in $(\Sigma_{v\in\mathcal{V}}(rr)(e)(e!_v)(rr!_v))^*$.

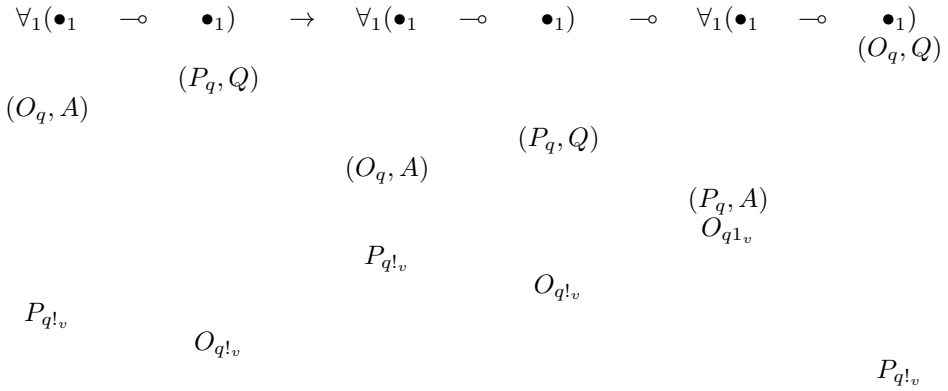
Instantiation of m-context games $B_1, \ldots, B_n$ in a n-context game A (corresponding to instantion of types for type variables) is by replacing each i-hole move (*and its response*) in A with a copy of B_i:

▶ **Definition 15.** For a n-context game A and well-opened m-context games $B_1, \ldots, B_n$, the m-context game $A[B_1, \ldots, B_n] = (A \backslash \bigcup\{\{u, u!_v \mid v \in \mathcal{V}\} \mid \lambda_A(u) = i, 1 \leq i \leq n\}) \cup \bigcup\{u.B_i \mid \lambda_A(u) = i, 1 \leq i \leq n\}$.

Instantiation into strategies is defined by extending them to the instantiated games by playing *copycat* between the game substituted for a question and that substituted for its answer. (Essentially, the application of the operation defined in [17] to the stack.)

▶ **Definition 16.** Given a legal stack t on a game D, let $\mathsf{Cl}(D, t)$ be the set of 2-question-answer pairs $\{(q, a) \in \lambda_D^{-1}(\mathsf{Q}) \times \lambda_D^{-1}(\mathsf{A}) \mid \exists sa \sqsubseteq t.\mathsf{pending}(s) = q\}$. For a morphism $\sigma : A \to C$ in $\mathcal{G}_C(n)$ define $\sigma[B_1, \ldots, B_n] : A[B_1, \ldots, B_n] \to C[B_1, \ldots, B_n]$ to be the set of legal sequences t on $A[B_1, \ldots, B_n] \to C[B_1, \ldots, B_n]$ such that $t\restriction\overrightarrow{\forall}(A \to C) \in \sigma$ and:
If $s \sqsubseteq t$, $\lambda_{A\to C}(q) = i$ and $(q, a) \in \mathsf{Cl}(\overrightarrow{\forall}(A \to C), \mathsf{stack}(s))$ then $\mathsf{stack}(s)\restriction q = \mathsf{stack}(s)\restriction a$, where $t\restriction m$ is the restriction of t to suffixes of m. We use the Q/A-copycat condition to show that this is a deterministic strategy.

As an example of how instantiation combines "schematic" copycat behaviour derived from second-order bracketing with direct copying of 1-answers, consider the strategy on $\forall_1(\bullet_1 \multimap \bullet_1) \to \forall_1(\bullet_1 \multimap \bullet_1) \multimap \forall_1(\bullet_1 \multimap \bullet_1)$ obtained by instantiating $\forall_1(\bullet_1 \multimap \bullet_1)$ into the identity on $\forall_1(\bullet_1 \multimap \bullet_1)$. This contains the following play, in which the first 1-answer determines all of the remaining moves by the Q/A-copycat condition.



We now define indexed categories of context games for each combination of constraints on strategies, yielding a model of System F typing [21]. For each $n \in \mathbb{N}$, let $\mathcal{G}_C(n)$ be the

category in which objects are negative n-context games and morphisms from A to B are C-strategies on the closed game $\forall_1 \ldots \forall_n (A \to B)$. Composition and identity in each $\mathcal{G}_C(n)$ are defined as for $\mathcal{G}_C$: proof that this yields a category follows [17].

Let $\mathcal{I}$ be the category in which objects are positive natural numbers and morphisms from m to n are m-tuples of well-opened n-context games. Composition of $(B_0, \ldots, B_m)$ with $(A_1, \ldots, A_l)$ is $(A_1[B_1, \ldots, B_m], \ldots A_l[B_1, \ldots, B_m])$ and the identity on n is $(\bullet_1, \ldots, \bullet_n)$. This is a category with finite products (arithmetic sums) and is finitely generated by them from the object 1. We define an $\mathcal{I}$-indexed symmetric monoidal category with (specified) finite products: the functor $\breve{\mathcal{G}}_C$ from $\mathcal{I}^{op}$ into the category of symmetric monoidal categories which sends n to the category $\mathcal{G}_C(n)$ and $(B_0, \ldots, B_m) : n \to m$ to the instantiation functor $_[B_0, \ldots, B_m] : \mathcal{G}_C(m) \to \mathcal{G}_C(n)$.

Let $\breve{\mathcal{G}}_C^{+1}$ be the $\mathcal{I}$-indexed category which sends n to $\mathcal{G}_C(n+1)$ and $(B_1, \ldots, B_m)$ to $_[B_1, \ldots, B_m, \bullet_{n+1}]$, and $J : \breve{\mathcal{G}}_C \to \breve{\mathcal{G}}_C^{+1}$ the $\mathcal{I}$-indexed inclusion functor (i.e. $J_n : \mathcal{G}_C(n) \to \mathcal{G}_C^{+1}(n)$ is the inclusion of $\mathcal{G}_C(n)$ into $\mathcal{G}_C(n+1)$).

▶ **Proposition 17.** *J has an indexed right adjoint $\forall : \breve{\mathcal{G}}_C^{+1} \to \breve{\mathcal{G}}_C$.*

Proof. For any n-context game A, and $n+1$-context game B, $\forall_n(J_n(A) \to B) = A \to \forall_n B$. Hence there is a natural isomorphism between $\mathcal{G}_C(n+1)(J_n(A), B)$ and $\mathcal{G}_C(n)(A, \forall_n(B))$. These satisfy the *Beck-Chevalley* condition and so form an indexed adjunction (see [17]). ◀

The indexed category $\breve{\mathcal{G}}_C$ possesses the structure identified for simple games – i.e. well-opened (indexed) Cartesian products, exponentials and cofree commutative comonoids.

5 Semantics of Dynamic Binding

Dynamically bound variables may be considered as objects with two methods – *definition* (using `let`), and *invocation*. In similar fashion to the semantics of ground-type and higher-order references [4, 5], our semantics is based on representing such an object as a strategy on the game $\S A = !(!A \multimap \Sigma) \otimes !A \cong !((!A \multimap \Sigma) \& A)$ so that invocation corresponds to right-projection, and $\texttt{let}\ x = M\ \texttt{in}\ N : T$ corresponds to applying the left projection from this product to M, instantiating with $[\![T]\!]$ and applying to N.

So we interpret a term $x_1 : S_1, \ldots, x_n : S_n \vdash M : T$ as a morphisms from $\S[\![S_1]\!] \otimes \ldots \otimes \S[\![S_n]\!]$ to $[\![T]\!]$ in $\mathcal{G}_C$. To interpret λ-abstraction, we define a declaration strategy $\mathsf{dec}_A : !A \to \S A$ which dynamically connects definitions to invocations, so that abstraction corresponds to composition with the strategy $\mathsf{dec}_A : !A \to \S A$, followed by currying: $[\![\Gamma \vdash \lambda x^S.M : S \to T]\!] : [\![\Gamma]\!] \to [\![S \to T]\!] = \Lambda((\![\Gamma]\!] \otimes \mathsf{dec}_{[\![S]\!]}); [\![\Gamma, x : S \vdash M : T]\!])$

The dynamic binding strategy is very similar to the reference cell strategy defined in [5]. The main difference that it is dynamically innocent – connecting only "reads" and "writes" *in the current block*. Informally, Proponent plays as the generic identity $\top$ on Σ and if Opponent requests an invocation by opening a thread of $!A$ on the right, then Proponent responds by playing copycat with the last-opened definition thread *on the stack*, if there is one, and

playing copycat with the source copy of $!A$, otherwise. An example play:

$$
\begin{array}{ccccccc}
!A & \to & !(!A & \multimap & \Sigma) & \otimes & !A \\
 & & & & O_q & & \\
 & & & & P_q & & \\
 & & & & & & O_q \\
 & & P_q & & & & \\
 & & O_{q!_v} & & & & \\
 & & & & & & P_{q!_v} \\
 & & & & O_{q!_v} & & \\
 & & & & P_{q!_v} & & \\
 & & & & & & O_q \\
P_q & & & & & &
\end{array}
$$

5.1 A Categorical Model of Dynamic Binding

In order to define the commutative monoid $(!A, \delta_A, \epsilon_A)$ and the dynamic binding strategy formally, and establish their key properties (that the former is the cofree commutative comonoid on A in $\breve{\mathcal{G}}_C$, and that the latter connects invocation and definition correctly), we give a categorical decomposition of the tensor into an *action* $_ \oslash _$: of each monoidal category $\breve{\mathcal{G}}_C$ on its (wide) subcategory of *block-linear* morphisms.

▶ **Definition 18.** $\sigma : A \to B$ is *block-linear* if $\forall s \in \sigma$ if $s{\restriction}B$ is complete then $s{\restriction}A$ is complete.

In other words, σ responds to the initial 1-question move by either closing it (with a 1-answer) or playing a 1-question move in B; when this is closed, σ either closes the opening move or opens a new block in B with a 1-question, and so on. This property is possessed by the identity, and preserved by composition, so for each n we may define $\breve{\mathcal{L}}_C$ to be the subcategory of $\breve{\mathcal{G}}_C$ in which objects are well-opened games and morphisms are block-linear strategies. This has cartesian products (given by &). For any game B, the monoidal exponential restricts to a functor $B \multimap _ : \breve{\mathcal{L}}_C \to \breve{\mathcal{L}}_C$ with the following property.

▶ **Lemma 19.** *For any B, the functor $B \multimap _ : \breve{\mathcal{L}}_C \to \breve{\mathcal{L}}_C$ has right and left adjoints, which commute with $B \multimap _$ (and therefore coincide) – i.e. $B \multimap _$ has a dual in the monoidal category of endofunctors on $\breve{\mathcal{L}}_C$.*

Proof. For any well-opened A, let $A \oslash B = ll.A + er.B$ (i.e. equivalent to $B \multimap A$ except that Proponent/Opponent roles are not swapped in B). Then there are evident adjunctions:

$$\frac{\breve{\mathcal{L}}_C(A \oslash B, C)}{\breve{\mathcal{L}}_C(A, B \multimap C)} \qquad \frac{\breve{\mathcal{L}}_C(B \multimap A, C)}{\breve{\mathcal{L}}_C(A, C \oslash B)}$$

and a natural isomorphism $\gamma : B \multimap (A \oslash B') \cong (B \multimap A) \oslash B'$. ◀

In the terminology of [15], this commuting adjunction defines a *sequoid*, i.e.:

- $_ \oslash _ : \breve{\mathcal{L}}_C \times \breve{\mathcal{G}}_C \to \breve{\mathcal{L}}_C$ is a monoidal *action* of $(\breve{\mathcal{G}}_C, \otimes, I)$ on $\breve{\mathcal{L}}_C$ (a strong monoidal functor $_ \oslash _$ from $\breve{\mathcal{G}}_C$ into the category of endofunctors on $\breve{\mathcal{L}}_C$).
- There is a monoidal natural transformation $\omega : J_ \otimes _ \to J(_ \oslash _)$ (where $J : \breve{\mathcal{L}}_C \to \breve{\mathcal{G}}_C$ is the inclusion functor) which commutes with the adjunctions $\otimes \dashv \multimap$ and $\oslash \dashv \multimap$.

The sequoid has the further property of "sequential decomposability": by conflict-freeness, any legal sequence on $A \otimes B$ is a concatenation of sequences from $A \oslash B$ and $B \oslash A$, and so:

▶ **Lemma 20.** *For any well-opened games A, B, $\langle \omega_{A,B}, \theta; \omega_{B,A} \rangle : A \otimes B \to (A \oslash B) \& (B \oslash A)$ is an isomorphism (where θ is the symmetry isomorphism of $\otimes$).*

Observe that $!A$ is a *minimal invariant* for the endofunctor $J(A \oslash _) : \breve{\mathcal{G}}_C \to \breve{\mathcal{G}}_C$: $!A = (er)^*.ll.A = ll.A \cup er.(er)^*.ll.A = A \oslash !A$ and the identity on $!A$ is the least fixed point for the operation taking $f : !A \to !A$ to $A \oslash f$. Following [15], we may use this property to define the cofree commutative comonoid structure of $!A$:

▶ **Definition 21.** For a well-opened game A, the commutative comonoid $(!A, \delta_A, \epsilon_A)$ in $\breve{\mathcal{G}}_C$ is *defined* as follows:

- $\delta_A : !A \to !A \otimes !A$ is the least fixed point of the endofunction on $\breve{\mathcal{G}}_C(!A, !A \otimes !A)$ sending f to $!A \cong A \oslash !A \xrightarrow{\mathsf{id}_A \oslash f} A \oslash (!A \otimes !A) \cong (A \oslash !A) \oslash !A \cong !A \oslash !A \xrightarrow{\Delta_{!A}} !A \otimes !A$.
- $\epsilon_A : !A \to I$ is the terminal map into I.

We show that this is the cofree commutative comonoid on A by defining:

- der_A is the morphism $!A \cong A \oslash !A \xrightarrow{\mathsf{id}_A \oslash \epsilon_A} A \oslash I \cong A$
- given $f : U(B) \to A$, $f^\dagger : B \to !A$ is the least fixed point of the operation sending $g : B \to !A$ to $B \xrightarrow{\delta_B} B \otimes B \xrightarrow{f \otimes g} A \otimes !A \xrightarrow{\omega_{A,!A}} A \oslash !A \cong !A$.

▶ **Proposition 22.** $!A$ *is the* cofree commutative comonoid *on* A *in* $\breve{\mathcal{G}}_C$.

Like the reference cell strategy [15], we can define $\mathsf{dec}_A : !A \multimap !(A \multimap \Sigma) \otimes !A$ formally, using the sequoidal decomposition of the cofree commutative comonoid, based on the counit $\eta_B : B \to (A \multimap B) \oslash A$ of the adjunction $_ \oslash A \dashv A \multimap _$.

▶ **Definition 23.** Recall that for any object B, $!(!A \multimap B) \otimes !A$ is the Cartesian product of $!(!A \multimap B) \oslash !A$ and $!A \oslash !(!A \multimap B)$ in $\breve{\mathcal{G}}_C$. and Let $\mathsf{dec}_A : !A \to !(!A \multimap \Sigma) \otimes !A$ be the least fixed point of the map $\Phi : \breve{\mathcal{G}}_C(!A, !(!A \multimap \Sigma) \otimes !A) \to \breve{\mathcal{G}}_C(!A, !(!A \multimap \Sigma) \otimes !A)$ sending f to the pairing of:
$!A \cong A \oslash !A \xrightarrow{A \oslash f} A \oslash (!(A \multimap \Sigma) \otimes !A) \cong A \oslash (!A \otimes !(!A \multimap \Sigma)) \cong (A \oslash !A) \oslash !(!A \multimap \Sigma) \cong !A \oslash !(!A \multimap \Sigma)$ and
$!A \xrightarrow{\epsilon_A} I \xrightarrow{\top} \Sigma \xrightarrow{\eta_{\Sigma,!A}} (!A \multimap \Sigma) \oslash !A \xrightarrow{(!A \multimap \Sigma) \oslash f} (!A \multimap \Sigma) \oslash (!(!A \multimap \Sigma) \otimes !A) \cong !(!A \multimap \Sigma) \oslash !A$

From this definition, we derive the following key equations relating $\mathsf{invoke}_A : \S A \to A \oslash \S A = (\delta_{\S A}; \omega_{\S A, \S A}); (\pi_r; \epsilon_A \oslash \S A)$ and $\mathsf{define} : \S A \to (!A \multimap \Sigma) \oslash \S A = (\delta_{\S A}; \omega_{\S A, \S A}); (\pi_l; \epsilon_A \oslash \S A)$.

Invocation: $\mathsf{dec}_A; \mathsf{invoke}_A : !A \to A \oslash \S A = (\delta_{!A}; \omega_{!A,!A}); (\mathsf{der}_A \oslash \mathsf{dec}_A)$ (i.e. reading a dynamically assigned variable returns its value and leaves the stack unchanged).

Definition: $\mathsf{dec}_A; \mathsf{define}_A : !A \to (!A \multimap \Sigma) \oslash \S A = t_{!A}; \Lambda((\top \otimes \mathsf{dec}_A); \omega_{\Sigma, \S A}); \gamma_{!A, \Sigma, \S A}$ – i.e. composing with definition returns a method which updates the variable and returns $\top$.

6 Denotational Semantics of $\mathbf{PCF}^{\mathbf{2}}_{d}$

We interpret our type theory using the categorical structure identified in the previous section. Each type $X_1, \ldots, X_n \vdash T$ is interpreted as a well-opened n-context game $[\![T]\!]_{X_1,\ldots,X_n}$:

$[\![S \to T]\!]_\Theta = !\,[\![S]\!]_\Theta \multimap [\![T]\!]_\Theta \qquad [\![X_i]\!]_{X_1,\ldots,X_n} = \bullet_i \qquad [\![\forall X_n.T]\!]_\Theta = \forall_n [\![T]\!]_{\Theta, X_n}$

Interpretation of the types and constants of PCF as games and BDI-strategies follows the games models in [3, 12] – i.e. `nat` denotes the flat game $\underline{\mathbb{N}}$. Terms $x_1 : S_1, \ldots, x_n : S_n \vdash_\Theta M : T$ are interpreted as morphisms from $\S[\![S_1]\!]_\Theta \otimes \ldots \otimes \S[\![S_n]\!]_\Theta$ to $[\![\mathbf{T}]\!]_\Theta$ in $\breve{\mathcal{G}}_C$, defined using the categorical structure we have identified:

Invocation and definition correspond to right and (application of) left projection from §:

$[\![x_1, : T_1 \ldots, x_n : T_n \vdash x_i : T_i]\!]_\Theta : \S[\![T_1]\!]_\Theta \otimes \ldots \otimes \S[\![T_n]\!]_\Theta \to [\![T_i]\!]_\Theta = \pi_i; \pi_r; \mathsf{der}$

$[\![\Gamma \vdash \mathtt{let}\ x_i = M\ \mathtt{in}\ N : T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![T]\!]_\Theta$

$= \delta_{[\![\Gamma]\!]}; ((\delta_{[\![\Gamma]\!]}; ((\pi_i; \pi_l; \mathsf{der}) \otimes [\Gamma \vdash M : S]\!]_\Theta)); \mathsf{eval})[[\![T]\!]_\Theta] \otimes [\![\Gamma \vdash N : T]\!]_\Theta); \mathsf{eval}$

Abstraction and application composition with the strategy dec, with currying/application:
$[\![\Gamma \vdash \lambda x^S.M : S \to T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![S \to T]\!]_\Theta = \Lambda((\![\Gamma]\!]_\Theta \otimes \mathsf{dec}_{[\![S]\!]_\Theta}); [\![\Gamma, x : S \vdash M : T]\!]_\Theta)$
$[\![\Gamma \vdash M\ N : T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![T]\!]_\Theta = \delta_{[\![\Gamma]\!]}; ([\![\Gamma \vdash M : S \to T]\!]_\Theta \otimes [\![\Gamma \vdash N : S]\!]_\Theta); \mathsf{eval}$
Type abstraction and instantiation are interpreted using second-order structure:
$[\![\Gamma \vdash \Lambda X.M : \forall X.T]\!]_\Theta : [\![\Gamma]\!]_\Theta \to [\![\vdash \forall X.T]\!]_\Theta = \forall_n [\![\Gamma \vdash M]\!]_{\Theta, X_n}$
$[\![\Gamma \vdash M\{S\} : T[S/X]]\!]_\Theta : [\![\Gamma]\!]_\Theta [[\![\vdash S]\!]_\Theta] \to [\![T]\!]_\Theta [[\![S]\!]_\Theta] = [\![\Gamma \vdash M]\!]_\Theta [[\![S]\!]_\Theta]$
Given a well-typed term $x_1 : S_1, \ldots, x_k : S_k \vdash N : T$, and $\mathcal{E} = (x_1, M_1), \ldots, (x_k, M_k)$, we define the interpretation of a configuration of the operational semantics:
$[\![\mathcal{E}; N]\!] = ((\bot; \mathsf{dec}_{S_1}) \otimes \ldots \otimes (\bot; \mathsf{dec}_{S_n})); [\![\mathtt{let}\ x_1 = M_1\ \mathtt{in}\ \ldots \mathtt{let}\ x_k = M_k\ \mathtt{in}\ N]\!]$.
Using the equalities established for the the dynamic binding operation, and the soundness of β-equivalence in the semantics of second order types, we show:

▶ **Proposition 24** (Soundness). *If* $(M; \mathcal{E}) \Downarrow C; \mathcal{E}'$ *then* $[\![M; \mathcal{E}]\!] = [\![C; \mathcal{E}']\!]$.

As in [17], we establish *computational adequacy* via a sequence of approximating semantics: For each n, we define an operational semantics of PCF^2_d in which each definition may be called at most n times in a block – i.e. application and `let` push n copies of (a, M) onto the stack and invocation removes one copy. Reduction with respect to this operational semantics is size-reducing with respect to a simple measure and thus terminating.

We then show that this operational semantics is sound, and therefore adequate, with respect to a denotational semantics in which the dec strategy is replaced by its nth approximant. Adequacy of the semantics now follows: for any term $M : \mathtt{nat}$, $[\![M]\!] = \bigcup_{n \in \omega} [\![M]\!]_n$. Hence if $[\![M]\!] \neq \bot$, there exists n with $[\![M]\!]_n \neq \bot$ and so $M \Downarrow$ as required.

▶ **Proposition 25.** $[\![\mathcal{E}; M]\!] = [\![\mathtt{n}]\!]$ *implies* $M; \mathcal{E} \Downarrow \mathtt{n}; \mathcal{E}'$ *for some* $\mathcal{E}'$.

We prove a definability property for first-order types analogous to the corresponding results for the games models of PCF: finite definability holds at type T if every *compact* BDI-strategy σ on $[\![T]\!]$ is the denotation of a term of type T (a dynamically innocent strategy σ is compact with the respect to the inclusion order if and only if the set $\{\lceil s \rceil \mid s \in \sigma\}$ is finite). The proof uses a decomposition argument based on similar principles to the original definability proof for PCF [3, 12]; we use the categorical decomposition of the tensor and cofree commutative comonoid into the sequoid to axiomatize this in a similar style to [1].

▶ **Proposition 26.** *Finite definability holds at all first-order types.*

To extend our definability result from first-order to second-order types, the key step is to simplify the latter using the observation that the type $\mathsf{I} = \forall X.X \to X$ is *strongly generic*:

▶ **Lemma 27.** *For any type* $T(X)$*, there is a* definable retraction $\forall X.T \trianglelefteq T[\mathsf{I}/X]$ *(i.e. there are* PCF^2_d *terms which denote a retraction/section between the corresponding type-objects).*

Proof. The retraction of $[\![\forall X.T]\!]$ into $[\![T[\mathsf{I}/X]]\!]$ is the instantiation morphism. The corresponding section plays copycat between $[\![T[\mathsf{I}/X]]\!]$ and $[\![\forall X.T]\!]$ until Opponent asks a question q in $[\![T[\mathsf{I}/X]]\!]$ corresponding to a negative occurrence of X in T: Proponent gives the sole answer to this; if Opponent copies this move as an answer to a question corresponding to a positive occurrence of X in T, then Proponent plays the move corresponding to q (which is an answer) in $[\![\forall X.[\![T]\!]]\!]$.

We may define this in PCF^2_d by giving terms $x : X \vdash \mathsf{in}_T : T[X] \to T[\mathsf{I}/X]$ and $x : X \vdash \mathsf{proj}_T : T[\mathsf{I}/X] \to T[X]$ such that $\lambda y^{T[\mathsf{I}/X]}.\Lambda X.(\lambda x.\mathsf{proj}_T)\,\Omega_X$ denotes the required strategy – i.e. $[\![\lambda z^{\forall X.T}.\Lambda X.(\lambda x.\mathsf{proj}_T\, z\{\mathsf{I}\})\,\Omega_X]\!] = [\![\lambda z.z]\!]$. The free variable x of type X is used to dynamically associate positive occurrences and negative occurrences of X:

- $\mathsf{in}_X : X \to \mathsf{I} = \lambda y^X.\mathtt{let}\ x = y\ \mathtt{in}\ \Lambda Z.\lambda z^Z.z$, $\mathsf{proj}_X : \mathsf{I} \to X = \lambda y^{\mathsf{I}}.y\{X\}\ x$.
- $\mathsf{in}_T = \mathsf{proj}_T = \lambda x^T.x$ for T a ground type, or variable type other than X.
- $\mathsf{in}_{S\to T} = \lambda f^{S\to T}.\lambda y^{S[\mathsf{I}/X]}.\mathsf{in}_T\ (\mathsf{proj}_S\ y)$, $\mathsf{proj}_{S\to T} = \lambda f^{S[\mathsf{I}/X]\to T[\mathsf{I}/X]}.\lambda y^S.\mathsf{proj}_T\ (\mathsf{in}_S\ y)$
- $\mathsf{in}_{\forall Y.T} = \lambda y^{\forall Y.T}.\Lambda Y.(\mathsf{in}_T\ y\{Y\})$, $\mathsf{proj}_T = \Lambda y^{\forall Y.T[\mathsf{I}/X]}.\Lambda Y.(\mathsf{proj}_T\ y\{Y\})$ ◀

This result may be used to eliminate (almost) all quantifiers: define the *almost quantifier free* types by the grammar: $U ::= \mathtt{nat} \mid \mathsf{I} \mid U \to U$.

▶ **Proposition 28.** *Finite definability holds at almost quantifier free types.*

▶ **Lemma 29.** *Finite definability holds at all types.*

Proof. Using Lemma 27, we may show that every closed type T is a retract of an almost quantifier free type T'. Thus if the image of $\sigma : [\![T]\!]$ under this retraction is definable as a term $M : T'$, then σ is definable as $\mathsf{proj}_T\ M$. ◀

7 Computational Effects with Dynamic Binding: Idealized Algol

As we have noted, we are in a situation analogous to PCF in that our definability result is obtained by applying three constraints to strategies (dynamic innocence, first-order well-bracketing and determinacy), which may be relaxed singly and in combination to obtain models of PCF^2_d with side-effects (local state [4], non-local control [14] and non-determinism [9]), with definability results obtained via the *factorization* results described in *loc. cit.* We illustrate the case of integer state, describing a model of IA^2_d – second-order Idealized Algol with dynamic binding. This is an expressive and useful combination of types and effects. In particular, existential types may be used to encapsulate the state of an object, and so define classes in Idealized Algol [22].

We define IA^2_d to be the proper extension of PCF^2_d obtained from our type theory by taking the set B of base types to consist of the types $\mathtt{nat}$, $\mathtt{com}$ (commands) and $\mathtt{var}$ (integer references) and adding additional constants for sequential composition of commands ($\mathtt{skip} : \mathtt{com}$ and $\mathsf{seq}_T : \mathtt{com} \to T \to T$) and declaration, assignment and dereferencing of integer variables ($\mathtt{new}_i : (\mathtt{var} \to \mathtt{com}) \to \mathtt{com}$, $\mathtt{assign} : \mathtt{var} \to \mathtt{nat} \to \mathtt{var}$ and $\mathtt{deref} : \mathtt{var} \to \mathtt{nat}$). We use common syntactic sugar for sequential composition, declaration, assignment, etc.

The operational semantics of IA^2_d extends that of PCF^2_d with a set of location names, Loc (which may occur as canonical forms), and adding to configurations a *store* (heap) $\mathcal{S}$ – a set of bindings of location names to integers defining a partial function from loc to $\mathbb{N}$ – i.e. $\Downarrow$ becomes a relation between triples $M;\mathcal{E};\mathcal{S}$ and $C,\mathcal{E}',\mathcal{S}'$. We decorate each PCF^2_d-rule with a store and add the following rules:

$$\frac{M\ a;\mathcal{E};\mathcal{S}\cup\{(a,i)\}\Downarrow C;\mathcal{E}';\mathcal{S}'\cup\{(a,v)\}}{\mathtt{new}_i\ M;\mathcal{E};\mathcal{S}\Downarrow C;\mathcal{E}';\mathcal{S}'}\ a\notin\mathsf{dom}(\mathcal{S}) \qquad \frac{M;\mathcal{E};\mathcal{S}\Downarrow \mathtt{skip};\mathcal{E}';\mathcal{S}'}{\mathsf{seq}_T\ M;\mathcal{E};\mathcal{S}\Downarrow\lambda x^T.x,\mathcal{E}'',\mathcal{S}''}$$

$$\frac{M;\mathcal{E};\mathcal{S}\Downarrow a;\mathcal{E}';\mathcal{S}' \quad N;\mathcal{E};\mathcal{S}'\Downarrow \mathtt{n},\mathcal{E}'',\mathcal{S}''}{M:=N;\mathcal{E};\mathcal{S}\Downarrow\mathtt{skip};\mathcal{E}'';\mathcal{S}''[a\mapsto n]} \qquad \frac{M;\mathcal{E};\mathcal{S}\Downarrow a;\mathcal{E}';\mathcal{S}'}{!M;\mathcal{E};\mathcal{S}\Downarrow \mathtt{n};\mathcal{E}';\mathcal{S}'}\ \mathcal{S}'(a)=n$$

$\mathtt{com}$ denotes the flat game $\underline{\{*\}}$ (with a single 1-question and 1-answer), and $\mathtt{var}$ the game $\mathsf{var} = [\![\mathtt{nat}]\!]\&[\![\mathtt{com}]\!]^\omega$ (the cartesian product of the types of its methods – read from the cell, assign it with 0, assign with 1,...). The constants for sequential composition and assignment and dereferencing (right and left projection from var) all denote dynamically innocent strategies. The only constant which does not is $\mathtt{new}_i : (\mathtt{var} \to \mathtt{com}) \to \mathtt{com}$, which denotes the composition of $x : \mathtt{var} \vdash \lambda f.f\ x$ with the "reference cell" strategy $\mathsf{cell}_i : I \multimap !\mathsf{var}$ which returns the last value written to it (or its initial value, i) [4]. This is not dynamically

innocent (in fact, cell_i is not $*$-closed): each Proponent 1-answer hides the preceding Opponent 1-question, so the state of the cell is never visible to Proponent.

To establish the soundness of this interpretation, we define the denotation of a configuration relative to a state transformation. Let cell_i^j be the restriction of the cell strategy which starts in a state in which it is assigned with i and terminates in a state in which it is assigned with j. Given a well-typed term $a_1 : \mathtt{var}, \ldots, a_m : \mathtt{var}, x_1 : S_1, \ldots, x_n : S_n \vdash N : T$, and $\mathcal{E} = (x_{i1}, M_1), \ldots, (x_{ik}, M_k)$, we define $[\![\mathcal{E}; N]\!]_{\mathcal{S} \mapsto \mathcal{S}'} = (\mathsf{cell}_{\mathcal{S}(a_1)}^{\mathcal{S}'(a_1)} \otimes \ldots \otimes \mathsf{cell}_{\mathcal{S}(a_m)}^{\mathcal{S}'(a_m)}); [\![E; N]\!]$.

▶ **Proposition 30.** *$[\![\mathcal{E}; M]\!]_{\mathcal{S} \mapsto \mathcal{S}'} = [\![v]\!]$ if and only if $M; \mathcal{E}; \mathcal{S} \Downarrow v; \mathcal{E}'; \mathcal{S}'$ for some $\mathcal{E}'$.*

Terms $M, N : T$ of IA_d^2 are *observationally equivalent* ($M \simeq_T N$) if for all contexts $C[_ : T] : \mathtt{com}$, $C[M] \Downarrow$ if and only if $C[N] \Downarrow$. We now show that our model of IA_d^2 captures this equivalence (full abstraction), following precisely the arguments for full abstraction of the game semantics of Idealized Algol itself in [5], extended to the language without *bad variables* in [19]. First, we add to IA_d^2 and its games model a *bad-variable constructor* $\mathtt{mkvar} : (\mathtt{nat} \to \mathtt{com}) \to \mathtt{nat} \to \mathtt{var}$, which denotes the currying of the pairing $\langle\langle(\pi_l \otimes \underline{i}); \mathsf{eval} \mid i \in \omega, \pi_r\rangle : [\![\mathtt{nat} \to \mathtt{com}]\!] \& [\![\mathtt{nat}]\!] \to ([\![\mathtt{com}]\!]^\omega \& [\![\mathtt{nat}]\!])$. Then for any IA_d^2 type T:

▶ **Proposition 31.** *Every compact BD-strategy on $[\![T]\!]$ is denoted by a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$.*

Proof. σ may be *factorized* into the composition of a dynamically innocent strategy $\widehat{\sigma} : ![\![\mathtt{var}]\!] \to [\![T]\!]$ with the strategy $\mathsf{cell}_0 : 1 \to ![\![\mathtt{var}]\!]$, by using the reference cell to record the history of play (see [4]). As in loc. cit. we may easily extend the proof of definability for PCF_d^2 to establish that $\widehat{\sigma}$ is the denotation of a term $x : \mathtt{var} \vdash M : T$ of $\mathrm{IA}_d^2 + \{\mathtt{mkvar}\} - \{\mathtt{new}\}$, and thus σ is definable as $\mathtt{new}_0\,(\lambda x.M)$ ◀

A sequence t is 1-well-bracketed if both Proponent and Opponent obey the 1-well-bracketing condition – i.e. if $s(m!_v) \sqsubseteq t$ then $\mathsf{stack}(s) = s'm$ for some s'. Let $\mathsf{comp}(\sigma)$ be the set of complete, well-bracketed sequences in σ.

▶ **Theorem 32** (Full abstraction)**.** *$M \simeq N$ if and only if $\mathsf{comp}([\![M]\!]) = \mathsf{comp}([\![N]\!])$.*

Proof. The implication from left to right is a consequence of computational adequacy. For the converse, if $\mathsf{comp}([\![M : T]\!]) \neq \mathsf{comp}([\![N : T]\!])$ then without loss of generality there exists a complete sequence $s \in [\![M]\!]$ such that $s \notin [\![N]\!]$. Thus the strategy $\sigma : [\![T]\!] \to [\![\mathtt{com}]\!]$ which consists of even prefixes of $(qs*)^*$ distinguishes them – i.e. $[\![M]\!]; \sigma \neq \bot$ and $[\![N]\!]; \sigma = \bot$. By Proposition 28, σ is definable as a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$. We can extend this result to IA_d^2 without $\mathtt{mkvar}$ by following McCusker's analysis of good variables in Idealized Algol [19]. This uses the observation that any IA_d^2-definable object of $\mathtt{var}$ type which responds to a read request with a given value, must behave in the same way when presented with a write request for the same value, to show that any (denotations of) terms which may be distinguished by a term of $\mathrm{IA}_d^2 + \mathtt{mkvar}$ may be distinguished by a term of IA_d^2. This argument extends *mutatis mutandis* to IA_d^2: in other words, there is a $\mathtt{mkvar}$-free term $L : T \to \mathtt{com}$ such that $[\![\mathcal{L}\, M]\!] \neq \bot$ and $[\![L\, N]\!] = \bot$ and so $L\, M \Downarrow$ and $L\, N \not\Downarrow$ by adequacy. ◀

8 Conclusions and Further Directions

We have described games models for polymorphic languages with late binding of variables, with and without mutable state (IA_d^2 and PCF_d^2): the former fully abstract for closed terms. By relaxing further constraints (bracketing, determinacy) we may interpret further effects such as non-local control.

The definability result for PCF^2_d implies that a fully abstract model can be obtained by an intrinsic preorder collapse construction (as in the original semantics of PCF [12, 3]): the problem of characterising this preorder remains open. Our definability and full abstraction results are restricted to terms of closed type: terms with free variables introduce a form of the *bad variable problem* which remains to be solved.

Although they do not contain general references, both languages are still very expressive. However, they open the possibility of defining bounded fragments (e.g. by bounding the nesting of function calls as in [6]) in which programs may be interpreted as as regular or visibly pushdown automata and so used to verify properties of polymorphic programs such as observational equivalence.

References

1. S. Abramsky. Axioms for full abstraction and full completeness. In *Essays in Honour of Robin Milner*. MIT Press, 1997.
2. S. Abramsky and R. Jagadeesan. A game semantics for generic polymorphism. *Annals of Pure and Applied Logic*, 133(1):3–37, 2004.
3. S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.
4. S. Abramsky and G. McCusker. Linearity, Sharing and State: a fully abstract game semantics for Idealized Algol with active expressions. In P.W. O'Hearn and R. Tennent, editors, *Algol-like languages*. Birkhauser, 1997.
5. S. Abramsky, K. Honda and G. McCusker. A fully abstract games semantics for general references. In *Proceedings of LICS'98*. IEEE Press, 1998. doi:10.1109/LICS.1998.705669.
6. D. Ghica, A. S. Murawaki, and C.-H. L. Ong. Syntactic control of concurrency. In *Proceedings of ICALP'04*, number 3142 in LNCS. Springer, 2004.
7. J.-Y. Girard. *Interprétation functionelle et elimination des coupures de l'arithmètique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.
8. R. Harmer. *Games and Full Abstraction for Nondeterministic Languages*. PhD thesis, Imperial College London, 1999.
9. R. Harmer and G. McCusker. A fully abstract games semantics for finite non-determinism. In *Proceedings of the Fourteenth Annual Symposium on Logic in Computer Science, LICS'99*. IEEE Computer Society Press, 1998.
10. D. Hughes. Games and definability for System F. In *Proceedings of the Twelfth International syposium on Logic in Computer Science, LICS'97*. IEEE Computer Society Press, 1997.
11. J. M. E. Hyland. Game semantics. In *Semantics and Logics of Computation (Publications of the Newton Institute)*. Cambridge University Press, 1995.
12. J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. *Information and Computation*, 163:285–408, 2000.
13. Y. Lafont. *Logiques, catégories et machines*. PhD thesis, Université Paris 7, 1988.
14. J. Laird. Full abstraction for functional languages with control. In *Proceedings of LICS'97*. IEEE Computer Society Press, 1997. doi:10.1109/LICS.1997.614931.
15. J. Laird. A categorical semantics of higher-order store. In *Proceedings of CTCS'02*, number 69 in ENTCS. Elsevier, 2002.
16. J. Laird. Game semantics for a polymorphic programming language. In *Proceedings of LICS'10*. IEEE Press, 2010.
17. J. Laird. Game semantics for a polymorphic programming language. *Journal of the ACM*, 60(4), 2013.
18. G. McCusker. *Games and full abstraction for a functional metalanguage with recursive types*. PhD thesis, Imperial College London, 1996. Cambridge University Press.

19 G. McCusker. On the semantics of the bad-variable constructor in Algol-like languages. In *Proceedings of MFPS XIX*, ENTCS, 2003. To appear.

20 J. Mitchell and G. Plotkin. Abstract types have existential type. *ACM transactions on Programming Languages and Systems*, 10(3):470–502, 1988.

21 A. Pitts. Polymorphism is set-theoretic constructively. In D. Pitt, editor, *Proceedings of CTCS'88*, number 283 in LNCS. Springer, 1988.

22 U. Reddy. Objects and classes in Algol-like languages. *Information and Computation*, 172(1):63–97, 2002.

23 J. C. Reynolds. Towards a theory of type structure. In *Proceedings of the Programming Symposium, Paris 1974*, number 19 in LNCS. Springer, 1974.

24 J. C. Reynolds. The essence of Algol. In *Algorithmic Languages*, pages 345–372. North Holland, 1981.

High-Quality Synthesis Against Stochastic Environments*

Shaull Almagor[†1] **and Orna Kupferman**[2]

1 **School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel**

2 **School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel**

Abstract

In the classical synthesis problem, we are given a linear temporal logic (LTL) formula ψ over sets of input and output signals, and we synthesize a transducer that realizes ψ: with every sequence of input signals, the transducer associates a sequence of output signals so that the generated computation satisfies ψ. One weakness of automated synthesis in practice is that it pays no attention to the quality of the synthesized system. Indeed, the classical setting is Boolean: a computation satisfies a specification or does not satisfy it. Accordingly, while the synthesized system is correct, there is no guarantee about its quality. In recent years, researchers have considered extensions of the classical Boolean setting to a quantitative one. The logic LTL[$\mathcal{F}$] is a multi-valued logic that augments LTL with quality operators. The satisfaction value of an LTL[$\mathcal{F}$] formula is a real value in $[0, 1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification.

Decision problems for LTL become search or optimization problems for LTL[$\mathcal{F}$]. In particular, in the synthesis problem, the goal is to generate a transducer that satisfies the specification in the highest possible quality. Previous work considered the worst-case setting, where the goal is to maximize the quality of the computation with the minimal quality. We introduce and solve the stochastic setting, where the goal is to generate a transducer that maximizes the expected quality of a computation, subject to a given distribution of the input signals. Thus, rather than being hostile, the environment is assumed to be probabilistic, which corresponds to many realistic settings. We show that the problem is 2EXPTIME-complete, like classical LTL synthesis. The complexity stays 2EXPTIME also in two extensions we consider: one that maximizes the expected quality while guaranteeing that the minimal quality is, with probability 1, above a given threshold, and one that allows assumptions on the environment.

1998 ACM Subject Classification F.4.3 Formal Languages, B.8.2 Performance Analysis and Design Aids, F.1.1 Models of Computation

Keywords and phrases Stochastic and Quantitative Synthesis, Markov Decision Process

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.28

* A full version of the paper is available at `http://arxiv.org/abs/1608.06567`.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410, from the Israel Science Foundation (grant no. 1229/10), and from the US-Israel Binational Science Foundation (grant no. 2010431).

© Shaull Almagor and Orna Kupferman;
licensed under Creative Commons License CC-BY

1 Introduction

Synthesis is the automated construction of a system from its specification: given a linear temporal logic (LTL) formula ψ over sets I and O of input and output signals, we synthesize a finite-state system that *realizes* ψ [11, 19]. At each moment in time, the system reads a truth assignment, generated by the environment, to the signals in I, and it generates a truth assignment to the signals in O. Thus, with every sequence of inputs, the system associates a sequence of outputs. The system realizes ψ if all the computations that are generated by the interaction satisfy ψ.

One weakness of automated synthesis in practice is that it pays no attention to the quality of the synthesized system. Indeed, the classical setting is Boolean: a computation satisfies a specification or does not satisfy it. Accordingly, while the synthesized system is correct, there is no guarantee about its quality. This is a crucial drawback, as designers would be willing to give up manual design only if automated-synthesis algorithms return systems of comparable quality. In recent years, researchers have considered several extensions and variants of the classical setting of synthesis. One class of extensions stays in the Boolean setting. For example, in practice we can often make assumptions on the behavior of the environment. An assumption may be direct, say given by an LTL formula that restricts the set of possible sequences of inputs [7], or conceptual, say rationality from the side of the environment, which may have its own objectives [15], or a bound on the size of the environment and/or the generated system [21, 16]. Another class of extensions moves to a quantitative setting, where a specification may have different satisfaction values in different systems. For example, in [3], the input to the synthesis problem includes also Mealy machines that grade different realizing systems. As another example, in [20], the specification formalism is the multi-valued logic LTL[$\mathcal{F}$], which augments LTL with quality operators. The satisfaction value of an LTL[$\mathcal{F}$] formula is a real value in $[0, 1]$, where the higher the value is, the higher is the quality in which the computation satisfies the specification. The synthesis algorithm then seeks systems of the highest possible quality. A quantitative approach can be taken also with Boolean specifications and involves a probabilistic view: the environment is assumed to generate input sequences according to some probability distribution. Then, instead of requiring the system to satisfy the specification in all computations generated by the environment, we measure the probability with which this happens [17].

Combining the multi-valued approach with the probabilistic one has led to the use of Markov Decision Processes (MDPs). Indeed, MDPs are a clean mathematical model that allows the analysis of quantitative objectives in a probabilistic environment. The intricacy of MDPs has led, in turn, to a plethora of works on synthesis with various constraints and reward models (e.g. [2, 6, 8, 10, 12, 1]). The starting point of these works is the MDPs. This is puzzling, as while MDPs offer a very clean framework for the analysis, they do not serve as a specification formalism. Thus, the crucial step of actually obtaining the MDPs is missing.

In this work, we consider *stochastic high-quality synthesis*, which combines the multi-valued approach with the probabilistic one. We build on known techniques for MDPs, and still keep the specification formalism accessible to designers. The specification is given by an LTL[$\mathcal{F}$] formula, the environment is assumed to be probabilistic, and we seek a system that maximizes the expected satisfaction value. To explain the setting better, let us first review shortly LTL[$\mathcal{F}$]. The linear temporal logic LTL[$\mathcal{F}$] extends LTL with an arbitrary set $\mathcal{F}$ of functions over $[0, 1]$. Using the functions in $\mathcal{F}$, a specifier can formally and easily prioritize the different ways of satisfaction. The logic LTL[$\mathcal{F}$] is really a family of logics, each parameterized by a set $\mathcal{F} \subseteq \{f : [0, 1]^k \to [0, 1] \mid k \in \mathbb{N}\}$ of functions (of arbitrary arity)

over $[0,1]$. For example, as in earlier work on multi-valued extensions of LTL (c.f., [13]), the set $\mathcal{F}$ may contain the $\min\{x,y\}$, $\max\{x,y\}$, and $1-x$ functions, which are the standard quantitative analogues of the $\wedge$, $\vee$, and $\neg$ operators. The novelty of LTL[$\mathcal{F}$] is the ability to manipulate values by arbitrary functions. For example, $\mathcal{F}$ may contain the quantitative operator $\triangledown_\lambda$, for $\lambda \in [0,1]$, which tunes down the quality of a sub-specification. Formally, the satisfaction value of the specification $\triangledown_\lambda \varphi$ is the multiplication of the satisfaction value of φ by λ. Another useful operator is the weighted-average function $\oplus_\lambda$. There, the satisfaction value of the formula $\varphi \oplus_\lambda \psi$ is the weighted (according to λ) average between the satisfaction values of φ and ψ. This enables the quality of the system to be an interpolation of different aspects of it. As an example, consider the LTL[$\mathcal{F}$] formula $\varphi = \mathsf{G}(\mathit{req} \rightarrow (\mathit{grant} \oplus_{\frac{2}{3}} \mathsf{X}\mathit{grant}))$. The formula specifies the fact that we want requests to be granted immediately and the grant to hold for two transactions. When this always holds, the satisfaction value is $\frac{2}{3} + \frac{1}{3} = 1$. We are quite okay with grants that are given immediately and last for only one transaction, in which case the satisfaction value is $\frac{2}{3}$, and less content when grants arrive with a delay, in which case the satisfaction value is $\frac{1}{3}$.

Consider a system that receives requests and generates grants and consider a specification ψ that has φ as defined above as a sub-formula. Other sub-formulas of ψ may require the system to generate as few grants as possible, say with $\varphi' = (\mathsf{FG}(\neg \mathit{req})) \rightarrow (\mathsf{G}\neg(\mathit{grant} \wedge \mathsf{X}\mathit{grant}))$. That is, if requests eventually stop arriving, then there cannot be two successive grants. The specification ψ cannot be realized with satisfaction value 1, as the system does not know in advance whether requests eventually stops arriving. Therefore, in order to get a satisfaction value above 0 in the subformula φ', the system must not generate two successive grants, bounding the satisfaction value of the subformula φ by $\frac{2}{3}$. If, however, the input signals are distributed so that *req* may hold with a positive probability at each moment in time, then the probability that an input sequence satisfies $\mathsf{FG}(\neg \mathit{req})$ is 0, causing φ' to be satisfied (that is, to have satisfaction value 1) with probability 1. Accordingly, under this assumption, a system that grants requests immediately and for two transactions has expected satisfaction value 1.

Formally, one can measure the quality of a system $\mathcal{S}$ with respect to an LTL[$\mathcal{F}$] specification taking three approaches. In the *worst-case approach*, the environment is assumed to be hostile and we care for the minimal satisfaction value of some computation of $\mathcal{S}$. In the *almost-sure approach*, the environment is assumed to be stochastic and we care for the maximal satisfaction value that is generated with probability 1. Then, in the *stochastic approach*, the environment is assumed to be stochastic and we care for the expected satisfaction value of the computations of $\mathcal{S}$, assuming some given distribution on the inputs sequences.

▶ **Example 1.** Consider a battery-replacement controller for a certain hardware. A computation of the hardware lasts k steps. Some steps during the execution are *stations*, in which the battery can be replaced. For example, the hardware may be an electric car whose battery can only be replaced at charging stations. The controller should decide at which stations it replaces the battery. On the one hand, it is wasteful to replace the battery early. On the other hand, the occurrence of stations is random, and the controller does not know whether stations are going to be encountered in the future.

Since it is wasteful to replace the battery early, the specification states that replacing it in step $1 \leq t \leq k$ lowers the satisfaction value to t/k. Missing, however, all stations incurs satisfaction value 0. We assume that each step is a station with probability $p \in [0,1]$.

Formally, the specification for the controller is over the sets $I = \{\mathit{station}\}$ and $O = \{\mathit{replace}\}$, and is a conjunction $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ of three LTL[$\mathcal{F}$] formulas (the abbreviation X^i stands for a sequence of i nested X (next) operators):

- $\varphi_1 = \mathsf{G}(\mathit{replace} \rightarrow \mathit{station})$, which requires that we only replace the battery in stations,
- $\varphi_2 = (\bigvee_{1 \leq t \leq k} \mathsf{X}^k \mathit{station}) \rightarrow (\bigvee_{1 \leq t \leq k} \mathsf{X}^k \mathit{replace})$, which states that the requirement to replace the battery needs to be satisfied only if at least one station has been encountered.
- $\varphi_3 = \bigwedge_{1 \leq t \leq k} \mathsf{X}^t(\neg \mathit{replace} \vee \nabla_{\frac{t}{k}} \mathit{replace})$, which lowers the satisfaction value to $\frac{t_0}{k}$, for the minimal step $1 \leq t_0 \leq k$ in which the battery is replaced.

In order to ensure a positive satisfaction value in the worst case, a transducer must replace the battery on the first station it encounters. Such a transducer guarantees a satisfaction value of $\frac{1}{k}$, but has expected satisfaction value of $(1-p)^k(1-\frac{1}{k}) + \frac{1}{k}$, which tends to 0 as k increases.

Trading-off the satisfaction value in the worst case for a higher expected satisfaction value, a controller may also replace the battery in later stations. For example, a transducer that replaces the battery only in the k-th step (if it is a station) has expected satisfaction value $(1-p)^k + p$. However, its satisfaction value in the worst case, in fact in $(1-p)$ of the computations, is 0.

In the full version we analyze the expected satisfaction value of a transducer that replaces the battery in the first station after position t, for $1 \leq t \leq k$, and show, for example, that a transducer that replaces the battery starting in position $\frac{k}{2}$ has an expected satisfaction value that tends to $\frac{1}{2}$ as $k \rightarrow \infty$, for every fixed $p \in (0,1)$. ◀

The worst case approach has been studied in [20], where it is shown how to synthesize, given φ, a system with a maximal worst-case satisfaction value. In this paper, we consider the two other approaches. We model a reactive system with sets I and O of input and output signals, respectively, by an I/O-transducer: a finite-state machine whose transitions are labeled by truth assignments to the signals in I and whose states are labeled by truth assignments to the signals in O. We define and solve the *stochastic high-quality synthesis* problem (SHQSyn, for short). The input to the problem is an LTL[$\mathcal{F}$] formula φ over $I \cup O$, and we seek an I/O-transducer that maximizes the expected satisfaction value of a computation, under a given distribution of the inputs. We show that the maximal expected satisfaction value is always attained by a finite-state transducer, and that computing such a transducer takes time that is doubly-exponential in φ, thus the problem is not more complex than the synthesis problem for LTL.

We continue to study two extensions of the SHQSyn problem. In the first extension, we add a lower bound on the satisfaction value that should be attained almost surely. Formally, the input to the *SHQSyn with threshold* problem is an LTL[$\mathcal{F}$] formula φ and a threshold $t \in [0,1]$, and we seek a transducer that maximizes the expected satisfaction value of φ, but such that the satisfaction value of φ in all its computations is at least t with probability 1. As we show, adding this restriction may lower the expected value. Also, our solution to the SHQSyn with threshold problem generalizes high-quality synthesis in the almost-sure approach, which we solve too. This approach has been studied for MDPs in [10, 12]. We show that while we can readily apply the existing solutions, the fact that our original specification is an LTL[$\mathcal{F}$] formula allows us to obtain slightly better solutions, with simpler analysis.

The second extension is the quantitative analogue of synthesis with environment assumptions. As discussed above, adding assumptions on the environment is a useful extension in the Boolean setting [7, 18]. In the *SHQSyn with environment assumption* problem we get as input an LTL[$\mathcal{F}$] formula φ and an environment assumption ψ, given by means of an LTL formula, and we seek a transducer that maximizes the expected satisfaction value of φ in computations that satisfy ψ. We note that the ability to reason about the quality of satisfaction in the presence of environment assumptions suggests a quantitative solution to

challenges that appear already in the Boolean setting. For example, in [4], the authors study the annoying phenomenon of systems realizing a specification by causing the assumption to fail. They suggest a synthesis algorithm that increases the cooperation between the system and its environment. Using LTL[$\mathcal{F}$], we can associate such a cooperation with high quality. We show that both extensions, of threshold and assumptions, as well as their combination, do not increase the complexity of the synthesis problem.

From a technical perspective, solving the Boolean synthesis problem amounts to translating an LTL formula to a deterministic parity automaton (DPW), viewing this automaton as a two-player parity game in which the system plays against the environment, and finding a winning strategy for the system. When the environment is assumed to be stochastic, the two-player game becomes a Markov decision process (MDP) with a parity objective. Such MDPs were extensively studied in [6, 8]. In order to handle the quantitative satisfaction values of LTL[$\mathcal{F}$], we translate an LTL[$\mathcal{F}$] formula φ to a set of DPWs associated with the different possible satisfaction values of φ. From the latter we obtain a mean-payoff MDP. We show that a transducer that attains the maximal expected satisfaction value is embodied in this MDP, and can be found in polynomial time. The analysis of the MDP is based on a search for *controllably win recurrent* states [8]. Adding a threshold $t \in [0, 1]$, the strategies of the MDP are restricted to those that guarantee that the computation reaches, with probability 1, end components that correspond to accepting runs of DPWs associated with satisfaction values above t.

Finally, in order to handle environment assumptions, we need to maximize the conditional expected satisfaction value, given the assumption. Maximizing conditional expectation is notoriously difficult, as, unlike unconditional expectation, it is not a linear objective. Thus, it is not susceptible to linear optimization techniques, which are the standard approach to find maximizing strategies in MDPs. In our solution, we compose the MDP with the DPW for the assumption, which enables us to adopt techniques used in the context of conditional probabilities in MDPs [2]. Intuitively, we add to the MDP transitions that "redistribute" the probability of computations that do not satisfy the assumption. In both cases, the size of the analyzed MDP stays doubly exponential in φ (and the assumption, in the latter case), and the required transducer is embodied in it.

Due to lack of space, some proofs are omitted and can be found in the full version, in the authors' home pages.

2 Preliminaries

2.1 Automata and Transducers

A (deterministic) *pre-automaton* is a tuple $\langle \Sigma, Q, q_0, \delta \rangle$, where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, and $\delta : Q \times \Sigma \nrightarrow Q$ is a (partial) transition function. A run of the pre-automaton on a word $w = \sigma_1 \cdot \sigma_2 \cdots \in \Sigma^\omega$ is a sequence of states $q_0, q_1, q_2, \ldots$ such that $q_{j+1} = \delta(q_j, \sigma_{j+1})$ for all $j \geq 0$. Note that since δ is deterministic, the pre-automaton has at most one run on each word.

A *deterministic parity automaton* (DPW, for short) is $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where $\langle \Sigma, Q, q_0, \delta \rangle$ is a pre-automaton, δ is a total function, and $\alpha : Q \to \{1, ..., d\}$ is an acceptance condition that maps states to ranks. The maximal rank d is the *index* of $\mathcal{A}$. For a run $r = q_0, q_1, q_2, \ldots$ of $\mathcal{A}$, let $\inf(r)$ be the set of states that occur in r infinitely often. Formally, $\inf(r) = \{q : q_j = q \text{ for infinitely many } j \geq 0\}$. The run r is accepting if the maximal rank of a state in $\inf(r)$ is even. Formally, $\max_{q \in \inf(r)}\{\alpha(q)\}$ is even. A word $w \in \Sigma^\omega$ is accepted by $\mathcal{A}$ if the run of $\mathcal{A}$ on w is accepting. The language of $\mathcal{A}$, denoted

$L(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

For finite sets I and O of input and output signals, respectively, an *I/O transducer* is $\mathcal{T} = \langle I, O, Q, q_0, \delta, \mu\rangle$, where $\langle 2^I, Q, q_0, \delta\rangle$ is a pre-automaton, and $\mu : Q \to 2^O$ is a labeling function on the states. Intuitively, $\mathcal{T}$ models the interaction of an environment that generates at each moment in time a letter in 2^I with a system that responds with letters in 2^O. Consider an input word $w = i_0 \cdot i_1 \cdots \in (2^I)^\omega$ and let $q_0, q_1, \ldots$ be the run of $\mathcal{T}$ on w. The *output* of $\mathcal{T}$ on w is then $o_1, o_2, \ldots \in (2^O)^\omega$, where $o_j = \mu(q_j)$ for all $j \geq 1$. Note that the first output assignment is that of q_1, thus $\mu(q_0)$ is ignored. This reflects the fact that the environment initiates the interaction. The *computation of $\mathcal{T}$ on w* is then $\mathcal{T}(w) = i_0 \cup o_1, i_1 \cup o_2, \ldots \in (2^{I \cup O})^\omega$.

2.2 Markov Chains and Markov Decision Processes

A *Markov chain* (MC, for short) $\mathcal{M} = \langle S, s_0, P\rangle$ consists of a finite or countably-infinite state space S, an initial state $s_0 \in S$, and a stochastic transition function $P : S \times S \to [0, 1]$. That is, for all $s \in S$, we have $\sum_{s' \in S} P(s, s') = 1$. Intuitively, when a run of $\mathcal{M}$ is in state s, then it moves to state s' with probability $P(s, s')$. A *run* of $\mathcal{M}$ is a finite or infinite sequence $s_0, s_1, s_2, ...$ of states that starts in s_0. The MC $\mathcal{M}$ induces a probability space on finite runs. Consider a finite run $r = s_0, s_1, ..., s_k$. We define $\Pr(r) = \prod_{i=1}^{k-1} P(s_i, s_{i+1})$. Thus, the probability of a finite run is the product of the probabilities of its transitions. Let $\mathrm{Cone}(r)$ be the set of all infinite runs that start with r. The MC $\mathcal{M}$ induces a probability space over the set of infinite runs of $\mathcal{M}$ that are generated by the cylinder sets $\mathrm{Cone}(r)$, for finite runs r. Formally, for every $r \in S^*$, we have $\Pr(\mathrm{Cone}(r)) = \Pr(r)$.

An *ergodic component* of $\mathcal{M}$ is a strongly connected component of $\mathcal{M}$ from which no other component is reachable. Formally, it is a set $C \subseteq S$ such that for every $s, t \in C$ there exist a path $s_1, s_2, ..., s_k$ of states in C such that $s_1 = s$, $s_k = t$, and $P(s_j, s_{j+1}) > 0$ for every $1 \leq j \leq k$. In addition, for every $s \in C$ and $t \notin C$, it holds that $P(s, t) = 0$. Let $\mathcal{C}$ be the set of maximal (w.r.t. containment) ergodic components of $\mathcal{M}$. We associate with $\mathcal{M}$ an *ergodic reachability probability* $\rho : \mathcal{C} \to [0, 1]$ such that $\rho(C)$ is the probability that a run of $\mathcal{M}$ reaches (and therefore remains forever in) C.

A *Markov decision process* (MDP) is $\mathcal{M} = \langle S, s_0, (A_s)_{s \in S}, \mathrm{P}, \gamma\rangle$, where S is a finite set of states, $s_0 \in S$ is an initial state, and A_s is a finite set of actions that are available in state $s \in S$. Let $A = \bigcup_{s \in S} A_s$. Then, $\mathrm{P} : S \times A \times S \nrightarrow [0, 1]$ is a (partial) stochastic transition function: for every two states $s, s' \in S$ and action $a \in A_s$, we have that $\mathrm{P}(s, a, s')$ is the probability of moving from s to s' when action a is taken. Accordingly, for every $s \in S$ and $a \in A_s$, we have $\sum_{s' \in S} \mathrm{P}(s, a, s') = 1$. Finally, $\gamma : S \to \mathbb{R}$ is a reward function on the states.

An MDP can be thought of as a game between a player, who chooses the action to be taken in each state, and nature, which stochastically chooses the next state according to the transition probabilities. The goal of the player is to maximize the average reward along the generated run in the MDP. We now formalize this intuition.

A *strategy* for the player in an MDP $\mathcal{M}$ (a strategy for $\mathcal{M}$, in short) is a function $f : S^+ \to A$ that suggests to the player an action to be taken given the history of the game so far. The strategy should suggest an available action, thus $f(s_0, \ldots, s_n) \in A_{s_n}$. A strategy is *memoryless* if it depends only on the current state. We can describe a memoryless strategy by $f : S \to A$, where again, $f(s) \in A_s$.

Given a strategy f, we can obtain from $\mathcal{M}$ the MC $\mathcal{M}_f = \langle S^+, s_0, P_f\rangle$ in which the choice of actions is resolved according to f. Formally, if $u, u' \in S^+$ are such that there are $t \in S^*$ and $s, s' \in S$ such that $u = t \cdot s$ and $u' = t \cdot s \cdot s'$, then $P_f(u, u') = \mathrm{P}(s, f(t \cdot s), s')$. Otherwise, $P_f(u, u') = 0$. Note that $\mathcal{M}_f$ has an infinite state space. If f is memoryless, we

can simplify the construction, and define $\mathcal{M}_f = \langle S, s_0, P_f \rangle$ with $P_f(s, s') = \mathrm{P}(s, f(s), s')$.

An *end component* in an MDP $\mathcal{M}$ is a set $C \subseteq S$ such that there exist action sets $(B_s)_{s \in S}$ with $B_s \subseteq A_s$ for every $s \in S$, and for every $s, t \in C$, there exists a path $s_1, s_2, ..., s_k$ of states in C such that $s_1 = s$, $s_k = t$ and there exist actions $a_1, ..., a_{k-1}$ such that $\mathrm{P}(s_j, a_i, s_{j+1}) > 0$ and $a_i \in B_{s_j}$ for every $1 \leq j \leq k$. In addition, for every $s \in C$ and $a \in B_s$ it holds that $\sum_{t \in C} \mathrm{P}(s, a, t) = 1$. Intuitively, an end component is a strongly-connected component in the MDP graph that nature cannot force to leave. Equivalently, $\mathcal{M}$ has a strategy to stay forever in C. Indeed, it is not hard to see that C is an end component iff there is some strategy f for $\mathcal{M}$ such that C is an ergodic component of $\mathcal{M}_f$.

The *value* $\mathrm{val}_{\mathcal{M}}(f)$ (we omit the subscript when $\mathcal{M}$ is clear from context) of a strategy f for $\mathcal{M}$ is the expected average reward of an infinite run in $\mathcal{M}_f$. Formally, for a run $r = s_0, s_1, s_2, \ldots$ of $\mathcal{M}_f$, we define $\gamma(r) = \liminf_{m \to \infty} \frac{1}{m} \sum_{j=0}^{m} \gamma(s_j)$, where for a state $s \in S^+$ of $\mathcal{M}_f$, the cost $\gamma(s)$ is induced by the last state of $\mathcal{M}$ in s. In the stochastic setting, we view each sequence of inputs, and hence also each run r and the reward on r, as a random variable. The *expected value* of a random variable is, intuitively, its average value, weighted by probabilities. Let $R_{\mathcal{M},f}$ be the random variable whose value is the reward on runs in $\mathcal{M}_f$. We define $\mathrm{val}_{\mathcal{M}}(f) = \mathbb{E}[R_{\mathcal{M},f}]$. The *value* $\mathrm{val}(\mathcal{M})$ of an MDP $\mathcal{M}$ is the maximal value of a strategy in $\mathcal{M}$. It is well known (see e.g. [14]) that $\mathrm{val}(\mathcal{M})$ can be attained by a memoryless strategy, which can be computed in polynomial time.

For technical reasons, we sometimes use variants of MDPs. A *pre-MDP* is an MDP with no reward function. A *parity MDP* is a pre-MDP with a parity acceptance condition $\alpha : S \to \{1, ..., d\}$. In a parity MDP, the goal of the player is to maximize the probability that the generated run satisfies the parity condition. Parity-MDPs were extensively studied in e.g. [9].

2.3 The logic LTL[$\mathcal{F}$]

The logic LTL[$\mathcal{F}$] is a multi-valued logic that extends the linear temporal logic LTL with an arbitrary set of functions $\mathcal{F} \subseteq \{f : [0,1]^k \to [0,1] : k \in \mathbb{N}\}$ called quality operators. For example, $\mathcal{F}$ may contain the maximum or minimum between the satisfaction values of subformulas, their product, and their average. This enables the specifier to refine the Boolean correctness notion and associate different possible ways of satisfaction with different truth values [20].

Let AP be a set of Boolean atomic propositions and let $\mathcal{F}$ be a set of function as described above. An LTL[$\mathcal{F}$] formula is one of the following:

- `True`, `False`, or p, for $p \in AP$.
- $f(\varphi_1, ..., \varphi_k)$, $\mathsf{X}\varphi_1$, or $\varphi_1 \mathsf{U} \varphi_2$, for LTL[$\mathcal{F}$] formulas $\varphi_1, \ldots, \varphi_k$ and a function $f \in \mathcal{F}$.

The semantics of LTL[$\mathcal{F}$] formulas is defined with respect to infinite computations over $2^{I \cup O}$. For a computation $\pi = \pi_0, \pi_1, \ldots \in (2^{I \cup O})^\omega$ and position $j \geq 0$, we use π^j to denote the suffix $\pi_j, \pi_{j+1}, \ldots$. The semantics maps a computation π and an LTL[$\mathcal{F}$] formula φ to the *satisfaction value* of φ in π, denoted $[\![\pi, \varphi]\!]$. The satisfaction value is in $[0,1]$ and is defined inductively as described in Table 1 below.

The logic LTL can be viewed as LTL[$\mathcal{F}$] for $\mathcal{F}$ that models the usual Boolean operators. For simplicity, we use the common such functions as abbreviation, as described below. In addition, we introduce notations for two useful quality operators, namely factoring and weighted average. Let $x, y, \lambda \in [0,1]$. Then,

- $\neg x = 1 - x$
- $x \vee y = \max\{x, y\}$
- $x \wedge y = \min\{x, y\}$

Table 1 The semantics of LTL[$\mathcal{F}$].

Formula	Satisfaction value	Formula	Satisfaction value
$[\![\pi, \texttt{True}]\!]$	1	$[\![\pi, f(\varphi_1, ..., \varphi_k)]\!]$	$f([\![\pi, \varphi_1]\!], ..., [\![\pi, \varphi_k]\!])$
$[\![\pi, \texttt{False}]\!]$	0	$[\![\pi, \mathsf{X}\varphi_1]\!]$	$[\![\pi^1, \varphi_1]\!]$
$[\![\pi, p]\!]$	$\begin{matrix}1 & \text{if } p \in \pi_0\\ 0 & \text{if } p \notin \pi_0\end{matrix}$	$[\![\pi, \varphi_1 \mathsf{U} \varphi_2]\!]$	$\max_{0 \le i < \lvert\pi\rvert} \{\min\{[\![\pi^i, \varphi_2]\!], \min_{0 \le j < i} [\![\pi^j, \varphi_1]\!]\}\}$

- $x \to y = \max\{1 - x, y\}$
- $\nabla_\lambda x = \lambda \cdot x$
- $x \oplus_\lambda y = \lambda \cdot x + (1 - \lambda) \cdot y$

▶ **Example 2.** Consider a scheduler that receives requests and generates grants and consider the LTL[$\mathcal{F}$] formula $\varphi = \varphi_1 \wedge \varphi_2$, with $\varphi_1 = \mathsf{G}(req \to \mathsf{X}(grant \oplus_{\frac{2}{3}} \mathsf{X}grant))$ and $\varphi_2 = \neg(\nabla_{\frac{3}{4}} \mathsf{G}\neg req)$. The satisfaction value of the formula φ_1 is 1 if every request is granted in the next cycle and the grant lasts for two consecutive cycles. If the grant lasts for only one cycle, then the satisfaction value is reduced to $\frac{2}{3}$ if it is the cycle right after the request, and to $\frac{1}{3}$ if it is the next one. In addition, the conjunction with φ_2 implies that if there are no requests, then the satisfaction value is at most $\frac{1}{4}$. The example demonstrates how LTL[$\mathcal{F}$] can conveniently prioritize different scenarios, as well as embody vacuity considerations in the formula. ◀

For an LTL[$\mathcal{F}$] formula φ, let $V(\varphi) = \{[\![\pi, \varphi]\!] : \pi \in (2^{AP})^\omega\}$. That is, $V(\varphi)$ is the set of possible satisfaction values of φ in arbitrary computations.

▶ **Theorem 3** ([20]). *Consider an* LTL[$\mathcal{F}$] *formula* φ.

- $|V(\varphi)| \le 2^{|\varphi|}$.
- *For every predicate* $\theta \subseteq [0, 1]$, *there exists a DPW* $\mathcal{A}_{\varphi,\theta}$ *such that* $L(\mathcal{A}_{\varphi,\theta}) = \{\pi : [\![\pi, \varphi]\!] \in \theta\}$. *Furthermore,* $\mathcal{A}_{\varphi,\theta}$ *has at most* $2^{2^{O(|\varphi|)}}$ *states and its index is at most* $2^{|\varphi|}$.

3 High-Quality Synthesis

Consider an I/O-transducer $\mathcal{T}$ and an LTL[$\mathcal{F}$] formula φ over $I \cup O$. Each computation of $\mathcal{T}$ may have a different satisfaction value for φ. We can measure the quality of $\mathcal{T}$ taking three approaches:

- *Worst-case approach:* The environment is assumed to be hostile and we care for the minimal satisfaction value of some computation of $\mathcal{T}$. Formally, $[\![\mathcal{T}, \varphi]\!]_w = \min\{[\![\mathcal{T}(w), \varphi]\!] : w \in (2^I)^\omega\}$. Note that no matter what the input sequence is, the specification φ is satisfied with value at least $[\![\mathcal{T}, \varphi]\!]_w$.
- *Almost-sure approach:* The environment is assumed to be stochastic and we care for the maximal satisfaction value that is generated with probability 1. Formally, given a distribution ν of $(2^I)^\omega$, we define $[\![\mathcal{T}, \varphi]\!]^\nu_a = \max\{v :$ there is W with $\nu(W) = 1$ and $[\![\mathcal{T}(w), \varphi]\!] \ge v$ for every $w \in W\}$. Note that the specification φ is satisfied almost surely with value at least $[\![\mathcal{T}, \varphi]\!]^\nu_a$.
- *Stochastic approach:* The environment is assumed to be stochastic and we care for the expected satisfaction value of the computations of $\mathcal{T}$, assuming some given distribution on the inputs sequences. Formally, let $X_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ be a random variable that assigns to each sequence $w \in (2^I)^\omega$ of input signals the value $[\![\mathcal{T}(w), \varphi]\!]$. Then, given a distribution ν of $(2^I)^\omega$, we define $[\![\mathcal{T}, \varphi]\!]^\nu_s = \mathbb{E}[X_{\mathcal{T},\varphi}]$, when the sequences in $(2^I)^\omega$ are sampled according to ν.

The worst case approach has been studied in [20], where it is shown how to find $[\![\mathcal{T},\varphi]\!]_w$ and how to synthesize, given φ, a transducer with a maximal worst-case satisfaction value. In this paper, we consider the stochastic approach. For simplicity, we consider environments with a uniform distribution on the input signals. That is, ν is such that at each moment in time, each input signal holds with probability $\frac{1}{2}$, thus the probability of each letter in 2^I is $\frac{1}{2^{|I|}}$ (see Remark 4). Since ν is fixed, we omit it from the notation and use $[\![\mathcal{T},\varphi]\!]_a$ and $[\![\mathcal{T},\varphi]\!]_s$.

▶ Remark 4 (On the choice of a uniform distribution). Recall that we consider a uniform distribution on the letters in 2^I. In practice, the distribution on the truth assignments to the input signals may be richer. In the general case, such a distribution can be given by an MDP.

Adjusting our setting and algorithms to handle such distributions involves only a small technical elaboration, orthogonal to the technical challenges that exist already in the setting of a uniform distribution. Accordingly, throughout the paper we assume a uniform distribution. In Section 7.2, we describe how our setting and algorithms are extended to the general case. ◀

▶ **Example 5.** Consider a hard-drive writing protocol that needs to finalize a write operation through some connection. The connection needs to be closed as soon as possible, to allow access to the drive. However, data may still arrive in the first two cycles, and if the connection is closed in the first cycle, then the data that arrives in the second cycle gets lost. The issue is that the decision as to whether to close the connection is made during the first cycle, before the protocol knows whether data is going to arrive in the second cycle. The specification that formulates the above scenario is over $I = \{data\}$ and $O = \{close\}$ and is $\varphi = ((\mathsf{X}data) \rightarrow \neg close) \wedge ((\neg \mathsf{X}data) \rightarrow close) \vee \nabla_{\frac{1}{2}} \mathsf{X}close)$.

That is, if data arrives in the second cycle, then we should not close the connection in the first cycle. In addition, if data does not arrive in the second cycle, we should close the connection in the first cycle – this would give us satisfaction value 1 in the second conjunct, but we may also close the connection only in the second cycle, which would guarantee a satisfaction value of 1 in the first conjunct, but would reduce the satisfaction value of the second conjunct to $\frac{1}{2}$ in cases data does not arrive in the second cycle.

Let $p \in [0,1]$ be the probability that data arrives in the second cycle. Consider a transducer $\mathcal{T}_1$ that closes the connection in the first cycle. With probability p, we have that $\mathsf{X}data$ holds, in which case φ has satisfaction value 0. Also, with probability $1-p$, we have that $\mathsf{X}data$ does not hold and the satisfaction value of φ is 1. Thus, the satisfaction value of φ is 0 in the worst case, and this is also the highest satisfaction value that $\mathcal{T}_1$ achieves with probability 1. On the other hand, the expected satisfaction value of φ in a computation of $\mathcal{T}_1$ is $p \cdot 0 + (1-p) \cdot 1 = 1-p$. Thus, $[\![\mathcal{T}_1,\varphi]\!]_w = [\![\mathcal{T}_1,\varphi]\!]_a = 0$, whereas $[\![\mathcal{T}_1,\varphi]\!]_s = 1-p$.

Consider now a transducer $\mathcal{T}_2$ that closes the connection only on the second cycle. With probability p, we have that $\mathsf{X}data$ holds, in which case the satisfaction value of φ is 1. Also, with probability $1-p$, we have that $\mathsf{X}data$ does not hold, in which case the satisfaction value of φ is $\frac{1}{2}$. Thus, now the satisfaction value of φ is $\frac{1}{2}$ in the worst case, and this is also the highest satisfaction value that $\mathcal{T}_2$ achieves with probability 1. On the other hand, the expected satisfaction value of φ in a computation of $\mathcal{T}_2$ is $p \cdot 1 + (1-p) \cdot \frac{1}{2} = \frac{1}{2}(1+p)$. Thus, $[\![\mathcal{T}_2,\varphi]\!]_a = [\![\mathcal{T}_2,\varphi]\!]_w = \frac{1}{2}$, whereas $[\![\mathcal{T}_2,\varphi]\!]_s = \frac{1}{2}(1+p)$.

To conclude, when $p \geq \frac{1}{3}$, in which case $\frac{1}{2}(1+p) \geq 1-p$, then $\mathcal{T}_2$ is superior to $\mathcal{T}_1$ in all the three approaches. When, however, $p < \frac{1}{3}$, then a designer that cares for the expected satisfaction value should prefer $\mathcal{T}_1$. ◀

3.1 The Achievability MDP of an LTL[$\mathcal{F}$] formula

In this section we develop the technical tool we are going to use for solving the high-quality synthesis problem in the stochastic approach.

Consider an LTL[$\mathcal{F}$] formula φ. Let $V(\varphi) = \{v_1, ..., v_n\}$, with $v_1 < ... < v_n \in [0, 1]$. By Theorem 3, we have that $n \leq 2^{|\varphi|}$. Also, for every $1 \leq i \leq n$, there is a DPW $\mathcal{A}_i$ such that $L(\mathcal{A}_i) = \{w : [\![w, \varphi]\!] = v_i\}$. Let $\mathcal{A}_i = \langle 2^{I \cup O}, Q^i, q_0^i, \delta^i, \alpha^i \rangle$. We construct the product pre-automaton $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ that subsumes the joint behavior of the DPWs. Formally, $\mathcal{A} = \langle 2^{I \cup O}, S, s_0, \mu \rangle$, where $S = Q^1 \times \ldots \times Q^n$, the initial state is $s_0 = \langle q_0^1, ..., q_0^n \rangle$, and for every state $s = \langle q_1, ..., q_n \rangle$ and $\sigma \in 2^{I \cup O}$, we have $\mu(s, \sigma) = \langle \delta^1(q_1, \sigma), ..., \delta^n(q_n, \sigma) \rangle$.

Every pre-automaton $\mathcal{B} = \langle 2^{I \cup O}, Q, q_0, \delta \rangle$ induces a pre-MDP $\mathcal{M}_\mathcal{B} = \langle Q, q_0, 2^O, \mathrm{P} \rangle$ in which for every two states $q, q' \in S$ and action $o \in 2^O$, we have $\mathrm{P}(q, o, q') = \frac{|\{i \in 2^I : \delta(q, i \cup o) = q'\}|}{2^{|I|}}$. That is, choosing action $o \in 2^O$ in state q, the MDP samples the possible inputs $i \in 2^I$ uniformly and moves to state $\delta(q, i \cup o)$. Consider a memoryless strategy $f : Q \to 2^O$ for $\mathcal{M}_\mathcal{B}$. The strategy f induces an I/O-transducer $\mathcal{T}[\mathcal{M}_\mathcal{B}, f] = \langle I, O, Q, q_0, \delta', \mu \rangle$ in which for every state $q \in Q$, we have $\mu(q) = f(q)$, and for all $i \in 2^I$, we have $\delta'(q, i) = \delta(i \cup \mu(q))$. Thus, the transducer has the same state space as $\mathcal{B}$, it lets f fix the labels of the states, and uses this label to complete the 2^I component of the alphabet to a letter in $2^{I \cup O}$.

Consider a parity acceptance condition α on the state space Q of $\mathcal{B}$. Using the notations of [9], a state $q \in Q$ in $\mathcal{M}_\mathcal{B}$ is *controllably win recurrent* (c.w.r., for short) if there exists an end component $U \subseteq Q$ such that $q \in U$, $\alpha(q) = \max_{p \in U} \{\alpha(p)\}$, and $\alpha(q)$ is even. That is, q has the maximal rank in U, and this rank is even. The end component U is referred to as a *witness* for q being c.w.r. Intuitively, a parity-MDP with a parity objective α has a strategy to win with probability 1 from all c.w.r. states. Moreover, if U is a witness for some c.w.r. state, then there exists a strategy to win with probability 1 from every state in U. If, however, a run of $\mathcal{M}_\mathcal{B}$ reaches, and stays forever, in an end component that does not have a c.w.r. state, then it is winning with probability 0.

Once we have defined the product pre-automaton $\mathcal{A}$, we construct an MDP $\mathcal{M}_\mathcal{A} = \langle S, s_0, 2^O, \mathrm{P}, \gamma \rangle$, with the following reward function. For a state $s = \langle q_1, ..., q_n \rangle$ of $\mathcal{M}_\mathcal{A}$, we say that a value $v_i \in V(\varphi)$ is *achievable* from s if there exists a c.w.r. state in $\mathcal{M}_{\mathcal{A}_i}$ with a witness U_i for which $q_i \in U_i$. Then, $\gamma(s) = \max\{v_i : v_i \text{ is achievable from } s\}$. Note that the way we have defined $\mathcal{A}$ guarantees that every state that is a part of some end component has at least one value v_i that is achievable from s. For states that are not in end components, we define the reward to be 0. Intuitively, $\gamma(s)$ is the highest satisfaction value that can be guaranteed with probability 1 from s. We refer to $\mathcal{M}_\mathcal{A}$ as the *achievability MDP for* φ.

This completes the construction of $\mathcal{M}_\mathcal{A}$. Note that every end component U consists of states with the same value v_U. Thus, every infinite run r of $\mathcal{M}$ eventually gets trapped in some end component U, implying that $\gamma(r) = v_U$. Indeed, the rewards along the states in the finite prefix of r that leads to U are averaged out. For an end-component U of $\mathcal{M}_\mathcal{A}$, let $U|_i$ be the projection of U on Q^i. Note that $U|_i$ is an end component in $\mathcal{A}_i$.

4 Synthesis Against a Stochastic Environment

In the *stochastic high-quality synthesis* problem (SHQSyn, for short), we get as input an LTL[$\mathcal{F}$] formula φ over sets I and O of input and output signals, and we seek an I/O-transducer that maximizes the expected value of a computation (under a uniform distribution of the inputs). Formally, we want to compute $\max_\mathcal{T} \{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$ and return the witness

transducer.[1]

We solve the SHQSyn problem by reasoning on the achievable MDP $\mathcal{M}_{\mathcal{A}}$. Consider a strategy f for $\mathcal{M}_{\mathcal{A}}$. Let $\mathcal{T}$ be the transducer induced from f, that is $\mathcal{T} = \mathcal{T}[M_{\mathcal{A}}, f]$. Recall the random variable $X_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ that maps $w \in (2^I)^\omega$ to $[\![\mathcal{T}(w), \varphi]\!]$. We define the random variables $Y_{\mathcal{T},\varphi} : (2^I)^\omega \to \mathbb{R}$ as follows. For every $w \in (2^I)^\omega$, we let $Y_{\mathcal{T},\varphi}(w)$ be the mean-payoff of the values along the run of $\mathcal{A}$ on $\mathcal{T}(w)$. Formally, let r be the run of $\mathcal{A}$ on $\mathcal{T}(w)$. Then, $Y_{\mathcal{T},\varphi}(w) = \gamma(r)$, where γ is the reward function of $\mathcal{M}_{\mathcal{A}}$. By definition, we have that $[\![\mathcal{T}, \varphi]\!]_s = \mathbb{E}[X_{\mathcal{T},\varphi}]$. Since $\mathcal{M}_{\mathcal{A}}$ is obtained by assuming a uniform distribution on the inputs, we have that $\mathbb{E}[Y_{\mathcal{T},\varphi}] = \mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f)$.

▶ **Theorem 6.** *Consider an* LTL[$\mathcal{F}$] *formula* φ*. Let* $\mathcal{M}_{\mathcal{A}}$ *be the achievability MDP for* φ*. For every value* $v \in [0,1]$*, there exists a strategy* f *in* $\mathcal{M}_{\mathcal{A}}$ *such that* $val_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$ *iff there exists an* I/O*-transducer* $\mathcal{T}$ *such that* $[\![\mathcal{T}, \varphi]\!]_s \geq v$*. Moreover, we can find in time polynomial in* $\mathcal{M}_{\mathcal{A}}$ *a memoryless strategy* f *such that* $[\![\mathcal{T}[\mathcal{M}_{\mathcal{A}}, f], \varphi]\!]_s$ *maximizes* $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$*.*

Proof. We start by proving that if there exists a transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!]_s \geq v$, then there exists a strategy f such that $\mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$. For this, we prove, in the full version, that $\mathbb{E}[X_{\mathcal{T},\varphi}] \leq \mathbb{E}[Y_{\mathcal{T},\varphi}]$. This is indeed sufficient, as we can then take f to be the strategy induced by $\mathcal{T}$.

For the converse implication, consider a strategy f in $\mathcal{M}_{\mathcal{A}}$ such that $\mathrm{val}_{\mathcal{M}_{\mathcal{A}}}(f) \geq v$. By [14], we can assume that f is memoryless. Let $\mathcal{T} = \mathcal{T}[\mathcal{M}_{\mathcal{A}}, f]$ be the transducer induced by f. In the full version, we show that there exists a transducer $\mathcal{T}'$ such that $\mathbb{E}[X_{\mathcal{T}',\varphi}] = \mathbb{E}[Y_{\mathcal{T},\varphi}]$, thus concluding the claim. ◀

We now proceed to show how to solve the SHQSyn problem.

▶ **Theorem 7.** *Solving the SHQSyn problem for* LTL[$\mathcal{F}$] *can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME complete.*

Proof. Consider an LTL[$\mathcal{F}$] formula φ. We want to find a transducer $\mathcal{T}$ that maximizes $[\![\mathcal{T}, \varphi]\!]_s$. Let $\mathcal{M}_{\mathcal{A}}$ be the achievability MDP for φ. By Theorem 6, we can find in time polynomial in $\mathcal{M}_{\mathcal{A}}$ a memoryless strategy f such that $[\![\mathcal{T}[\mathcal{M}_{\mathcal{A}}, f], \varphi]\!]_s$ maximizes $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s]\}$. Below we analyze the size of $\mathcal{M}_{\mathcal{A}}$. Let $|\varphi| = k$. By Theorem 3, we have that $n \leq 2^k$ and each $\mathcal{A}_i$ is of size at most $2^{2^{O(k)}}$. Thus, the size of $\mathcal{M}_{\mathcal{A}}$ is at most $(2^{2^{O(k)}})^{2^k} = 2^{2^{O(k)}}$, implying the doubly exponential upper bound.

A matching lower bound for the respective decision problem follows from the 2EXPTIME hardness of standard LTL synthesis. Note that in our setting one considers satisfaction with probability 1. Still, since the hardness proof for LTL synthesis considers the interaction between a system and its environment along a finite prefix of a computation (one that models the computation of a Turing machine that halts), it applies also for the stochastic setting. ◀

5 Adding an Almost-Sure Threshold

In this section we combine the stochastic and the almost-sure approaches. The *SHQSyn problem with a threshold* includes both an LTL[$\mathcal{F}$] formula φ and a threshold $t \in [0,1]$. The goal then is to maximize the expected satisfaction value of φ while guaranteeing that it is almost surely above t. Formally, given φ and t, we seek a transducer $\mathcal{T}$ that maximizes

$$\{[\![\mathcal{T}, \varphi]\!]_s : [\![\mathcal{T}, \varphi]\!]_a \geq t\}.$$

[1] A-priori, it is not clear that the maximum is attained. As we prove, however, this is in fact the case.

Note that there need not be a transducer $\mathcal{T}$ for which $[\![\mathcal{T},\varphi]\!]_a \geq t$, in which case the set is empty and we return no transducer. This is the multi-valued analogue of an unrealizable Boolean specification (except that here the user may want to try to reduce t). Note also that this sub-problem, of deciding whether the set is empty, amounts to solving the high-quality synthesis problem in the almost-sure approach. Finally, if the set is not empty, then we have to show, as in Section 4, that its maximum is indeed attained.

▶ **Example 8.** Consider a server sending messages over a noisy channel. At each cycle, the server sends a message and needs to decide whether to encode it so that error-correction can retrieve it in case the channel is noisy, or take a risk and send the message with no encoding. Encoding a message has some cost. We formulate the quality of each cycle by the specification ψ over $I = \{noise\}$ and $O = \{encode\}$, where $\psi = (\neg noise \wedge \neg encode) \vee \nabla_{\frac{3}{4}} encode$. Thus, each cycle has satisfaction value 1 if a message that is not encoded is sent over a non-noisy channel, and satisfaction value $\frac{3}{4}$ if a message is encoded. Note that otherwise (that is, when a message that is not encoded is sent over a noisy channel), the satisfaction value is 0. The factor $\frac{3}{4}$ in the LTL[$\mathcal{F}$] specification reflects the priorities of the designer as induced by the actual cost of encoding and of losing messages.

Recall that ψ specifies the quality of a single cycle. The quality of a full computation refers to its different cycles, and a natural thing to do is to take the average over the cycles we want to consider. Assume that a channel may be noisy only during the first four cycles. Then, the quality of a computation is $\varphi = (\psi \oplus_{\frac{1}{2}} \mathsf{X}\psi) \oplus_{\frac{1}{2}} (\mathsf{XX}\psi \oplus_{\frac{1}{2}} \mathsf{XXX}\psi)$.

Assume that the probability of a channel to be noisy in each of the first four cycles is p. Consider a transducer $\mathcal{T}_1$ that does not encode any message. The expected satisfaction value of ψ in each of the four cycles is then $(1-p)\cdot 1 + p\cdot 0 = 1-p$, hence $[\![\mathcal{T}_1,\varphi]\!]_s = 1-p$. On the other hand, the satisfaction value of ψ in a noisy cycle is 0, hence $[\![\mathcal{T}_1,\varphi]\!]_w = [\![\mathcal{T}_1,\varphi]\!]_a = 0$. Thus, if one does not care for a lower bound on the satisfaction value in the worst case, then by using $\mathcal{T}_1$ he gets an expected satisfaction value of $1-p$.

Suppose now that we want the satisfaction value to be above $\frac{1}{3}$ in the worst case. This can be achieved by a transducer $\mathcal{T}_2$ that encodes messages in two of the four cycles. Indeed, for the cycles in which a message is encoded, we get satisfaction value $\frac{3}{4}$, which is averaged with 0, namely the worst-case satisfaction value in the cycles in which a message is not encoded. Hence, $[\![\mathcal{T}_2,\varphi]\!]_w = [\![\mathcal{T}_2,\varphi]\!]_a = \frac{3}{4} \oplus_{\frac{1}{2}} 0 = \frac{3}{8} > \frac{1}{3}$. The expected satisfaction value of $\mathcal{T}_2$ is then $[\![\mathcal{T}_2,\varphi]\!]_s = \frac{3}{4} \oplus_{\frac{1}{2}} (1-p) = \frac{7}{8} - \frac{p}{2}$.

Finally, if we want to ensure satisfaction value $\frac{3}{4}$ in the worst case, then we can design a transducer $\mathcal{T}_3$ that encodes all the messages in the first four cycles. Now, $[\![\mathcal{T}_3,\varphi]\!]_w = [\![\mathcal{T}_3,\varphi]\!]_a = [\![\mathcal{T}_3,\varphi]\!]_s = \frac{3}{4}$.

It follows that for a small p, adding a threshold on the satisfaction value in the worst case reduces the expected satisfaction value. Indeed, when $p < \frac{1}{4}$, then $1-p > \frac{7}{8} - \frac{p}{2} > \frac{3}{4}$. When, however, $p \geq \frac{1}{4}$, then $\mathcal{T}_3$ is superior in the three approaches. ◀

In order to solve the SHQSyn problem with a threshold, we modify our solution from Section 3.1 as follows. We start by deciding whether there exists a transducer $\mathcal{T}$ such that $[\![\mathcal{T},\varphi]\!]_a \geq t$. For this, we construct, per Theorem 3, a DPW $\mathcal{A}_{\geq t} = \langle 2^{I\cup O}, Q^{\geq t}, q_0^{\geq t}, \delta^{\geq t}, \alpha^{\geq t}\rangle$ such that $L(\mathcal{A}_{\geq t}) = \{w : [\![w,\varphi]\!] \geq t\}$. Let $\mathcal{M}_{\geq t}$ be the parity-MDP induced from $\mathcal{A}_{\geq t}$. By [9], we can find the set of almost-sure winning states of $\mathcal{M}_{\geq t}$. If $q_0^{\geq t}$ is winning, then the required transducer exists, and in fact $\mathcal{M}_{\geq t}$ embodies all candidate transducers. We obtain a pre-automaton $\mathcal{A}'_{\geq t}$ from $\mathcal{A}_{\geq t}$ by restricting $\mathcal{A}_{\geq t}$ to winning states, and removing transitions from state $q \in Q^{\geq t}$ for every action $o \in 2^O$ such that there exists $i \in 2^I$ for which $\delta^{\geq t}(q, i \cup o)$ is not a winning state.

We proceed by constructing a product pre-automaton $\mathcal{A}$ that is similar to the one constructed in Section 3.1, except that takes t and $\mathcal{A}'_{\geq t}$ into account, as follows.

Let $\ell = \arg\min_i \{v_i : v_i \geq t\}$ be the minimal index such that $v_i \geq t$. We define $\mathcal{A} = \mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\geq t}$. That is, the product, defined as in Section 3.1, now contains only DPWs $\mathcal{A}_i$ for which $v_i \geq t$ and also contains $\mathcal{A}'_{\geq t}$. We obtain the MDP $\mathcal{M}_\mathcal{A}$ and set the reward function as in Section 3.1, taking into account only c.w.r. states from the automata $\mathcal{A}_\ell, ..., \mathcal{A}_n$. The component $\mathcal{A}'_{\geq t}$ is only used to restrict the actions of the MDP $\mathcal{M}_\mathcal{A}$. We refer to $\mathcal{M}_\mathcal{A}$ as the *t-achievability MDP for φ*.

We present an analogue to Theorem 6. The proof appears in the full version.

▶ **Theorem 9.** *Consider an* LTL[$\mathcal{F}$] *formula φ and a threshold $t \in [0,1]$. Let $\mathcal{M}_\mathcal{A}$ be the t-achievability MDP for φ. For every value $v \in [0,1]$, there exists a strategy f in $\mathcal{M}_\mathcal{A}$ such that $val_{\mathcal{M}_\mathcal{A}}(f) \geq v$ iff there exists an I/O-transducer $\mathcal{T}$ such that $[\![\mathcal{T}, \varphi]\!]_a \geq t$ and $[\![\mathcal{T}, \varphi]\!]_s \geq v$. Moreover, we can find in time polynomial in $\mathcal{M}_\mathcal{A}$ a memoryless strategy f such that $[\![\mathcal{T}[\mathcal{M}_\mathcal{A}, f], \varphi]\!]_s$ maximizes $\{\mathbb{E}[[\![\mathcal{T}, \varphi]\!]_s] : [\![\mathcal{T}, \varphi]\!]_a \geq t\}$.*

Since, by Theorem 3, the size of $\mathcal{A}_{\geq t}$ is doubly exponential in φ, then, by following considerations similar to those specified in the proof of Theorem 7, we conclude with the following.

▶ **Theorem 10.** *Solving the SHQSyn problem with a threshold for* LTL[$\mathcal{F}$] *can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME-complete.*

▶ Remark 11. In [10, 12], the authors solve the problem of deciding, given an MDP $\mathcal{M}$ and two thresholds v and t, whether there is a strategy f for $\mathcal{M}$ that guarantees value t almost surely, and has expected cost at least v. The solution can be directly applied to our setting. However, note that this solution only guarantees an expected cost of v, whereas our approach finds the optimal expected cost. ◀

▶ Remark 12. In the SHQSyn problem with a threshold, we use the formula φ both for the expectation maximization, and for the almost-sure threshold. Sometimes, it is desirable to decompose the specification into one part – ψ, which is a hard constraints and needs to be satisfied almost-surely above the threshold t, and another part – φ, which specifies a utility function with respect to which we would like to optimize [5, 12].

Our solution can be easily adapted to handle this setting. Indeed, in the construction of the t-achievability MDP, we replace $\mathcal{A}_{\geq t}$, with $\mathcal{B}_{\geq t}$, where $L(\mathcal{B}_{\geq t}) = \{w : [\![w, \psi]\!] \geq t\}$, and proceed with the described construction and the proofs. ◀

6 Adding Environment Assumptions

A common paradigm in Boolean synthesis is synthesis with environment assumptions [7, 18], where the input to the synthesis problem consists of a specification φ and an assumption ψ, and we seek a transducer that realizes φ under the assumption that the environment satisfies ψ. In this section we consider an analogue variant of the SHQSyn problem, where we are given an LTL[$\mathcal{F}$] specification φ and an LTL assumption ψ, and we seek a transducer that maximizes the expected satisfaction value of φ given that the environment satisfies the assumption ψ. Note that while the specification is quantitative, the assumption is Boolean.

▶ **Example 13.** Recall the message-sending server in Example 8, and assume that the channel can change its status (noisy/non-noisy) only every second cycle. We use this assumption in order to design improved transducers. Formally, the assumption is given by the LTL formula $\psi = (noise \leftrightarrow \mathsf{X} noise) \wedge \mathsf{XX}(noise \leftrightarrow \mathsf{X} noise)$.

The transducer $\mathcal{T}_4$ does not encode the first message, but checks whether the channel was noisy. If it was, the second message is encoded. We get that the expected satisfaction value of φ in $\mathcal{T}_4$ under the assumption is $(1-p+p\cdot\frac{3}{4}+(1-p)\cdot 1)/2 = 1-\frac{5}{8}p$, which is higher than $1-p = [\![\mathcal{T}_1,\varphi]\!]_s$ for every $p>0$. In addition, under the assumption we are guaranteed that the worst-case satisfaction value of $\mathcal{T}_4$ is at least $\frac{3}{8}$, unlike $\mathcal{T}_1$ (in case the channel is noisy, so only the second and fourth messages are encoded). Thus $\mathcal{T}_4$ is superior to $\mathcal{T}_1$ described in Example 8 in the three approaches (under the assumption).

Next, as in Example 8, if we want to ensure satisfaction value $\frac{3}{4}$ in the worst case, we can design a transducer $\mathcal{T}_5$ that works like $\mathcal{T}_4$, except that it always encodes the first and third messages. The expected satisfaction value of $\mathcal{T}_5$ under the assumption is $(\frac{3}{4}+p\cdot\frac{3}{4}+(1-p)\cdot 1)/2 = \frac{7}{8}-\frac{p}{8}$, which is higher than $\frac{3}{4} = [\![\mathcal{T}_3,\varphi]\!]_s$, for every $p\in[0,1]$.

Thus, under the assumption, it is possible to design transducers that increase the expected satisfaction value as well as the lower bound. ◀

Formally, in the SHQSyn problem with environment assumptions, we get as input an LTL[$\mathcal{F}$] formula φ over $I\cup O$, and an *environment assumption* ψ, which is an LTL formula over I such that $\Pr(\psi)>0$. That is, the probability of the event $\{w : w\models\psi\}\subseteq(2^I)^\omega$ is strictly positive. Recall that $X_{\mathcal{T},\varphi}$ is a random variable such that $X_{\mathcal{T},\varphi}(w) = [\![\mathcal{T}(w),\varphi]\!]$. We seek a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w\models\psi]$.

We start by citing a folklore lemma, whose proof can be found in the full version.

▶ **Lemma 14.** *Consider a random variable X. Let A,B be events such that $\Pr(A)>0$ and $\Pr(B)=0$. Then, $\mathbb{E}[X|A\cup B] = \mathbb{E}[X|A]$.*

Before proceeding, we note that if $\Pr(\psi)=1$, then we can proceed by dropping the assumption entirely. Indeed, it holds that $\Pr(\neg\psi)=0$, and by Lemma 14, we have that $\mathbb{E}[X_{\mathcal{T},\varphi}|w\models\psi] = \mathbb{E}[X_{\mathcal{T},\varphi}|(w\models\psi)\cup(w\models\neg\psi)] = \mathbb{E}[X_{\mathcal{T},\varphi}|(2^I)^\omega] = \mathbb{E}[X_{\mathcal{T},\varphi}]$. Thus, we henceforth assume that $0<\Pr(\psi)<1$.

As mentioned in Section 1, maximizing the conditional expectation directly is notoriously problematic, as, unlike unconditional expectation, it is not a linear objective. Thus, it is not susceptible to linear optimization techniques, which are the standard approach to find maximizing strategies in MDPs. Our solution is a modification of the construction from Section 3.1 in which we, intuitively, "redistribute" the probability of the input sequences that do not satisfy the assumption. We start by constructing a DPW $\mathcal{A}_\psi$ that accepts a word $w\in(2^I)^\omega$ iff $w\models\psi$. Note that the alphabet of $\mathcal{A}$ is 2^I. We think of this alphabet as $2^{I\cup O}$, where transitions simply ignore the 2^O component. In particular, the MDP $\mathcal{M}_{\mathcal{A}_\psi}$ is in fact an MC. We say that an ergodic component of $\mathcal{M}_{\mathcal{A}_\psi}$ is *rejecting* if the maximal rank that appears in it is odd. It is easy to see that a run in a rejecting ergodic component is accepting w.p. 0.

We then consider the automaton $\mathcal{A} = \mathcal{A}_\psi\times\mathcal{A}_1\times\ldots\times\mathcal{A}_n$, and obtain the MDP $\mathcal{M}_\mathcal{A} = \langle S, s_0, 2^O, \mathrm{P}, \gamma\rangle$ as described in Section 3.1. In particular, the reward function is as there, and the only change is the addition of the $\mathcal{A}_\psi$ component, which provides information about satisfaction of ψ. We refer to $\mathcal{M}_\mathcal{A}$ as the *conditional achievability MDP for φ given ψ*. Recall that for a strategy f, we have defined $R_{\mathcal{M}_\mathcal{A},f}$ as a random variable whose value is the reward on runs in $\mathcal{M}_\mathcal{A}$ with strategy f. Following the proof of Theorem 6, we then get the following.

▶ **Theorem 15.** *Consider an* LTL[$\mathcal{F}$] *formula φ and an environment assumption ψ. Let $\mathcal{M}_\mathcal{A}$ be the conditional achievability MDP for φ given ψ. For every value $v\in[0,1]$, there exists a strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_\mathcal{A},f}|w\models\psi]\geq v$ iff there exists an I/O-transducer*

$\mathcal{T}$ such that $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi] \geq v$. Moreover, if f is memoryless, then we can find in time polynomial in $\mathcal{M}_\mathcal{A}$ a memoryless strategy f such that $\mathbb{E}[X_{\mathcal{T}[\mathcal{M}_\mathcal{A},f],\varphi}|w \models \psi] \geq v$.

Theorem 15 enables us to reason about $\mathcal{M}_\mathcal{A}$, but we are still left with conditional expectations. To handle the latter, we follow a technique suggested in [2] and obtain from $\mathcal{M}_\mathcal{A}$ a new MDP $\mathcal{M}'_\mathcal{A} = \langle S, s_0, A, \mathrm{P}', \gamma\rangle$ as follows. A state $s = \langle q, q_1, ..., q_n\rangle$ of $\mathcal{M}_\mathcal{A}$ is called a *rejecting ergodic state* if its state q of $\mathcal{A}_\psi$ belongs to a rejecting ergodic component of $\mathcal{M}_{\mathcal{A}_\psi}$. Let $\mathscr{R} = \{s : s \text{ is a rejecting ergodic state}\}$.

For every state $s \in \mathscr{R}$ we set $\mathrm{P}'(s, a, s_0) = 1$. That is, whenever a rejecting ergodic component of $\mathcal{A}_\psi$ is reached, the MDP $\mathcal{M}'_\mathcal{A}$ deterministically resets back to s_0.

Intuitively, when a rejecting ergodic component of $\mathcal{A}_\psi$ is reached, then the probability of ψ being satisfied is 0. Thus, resetting "redistributes" the probability of ψ not being satisfied evenly. Below we formalize this intuition. The proofs can be found in the full version.

▶ **Lemma 16.** *Let $v \in \mathbb{R}$, and consider a memoryless strategy g in $\mathcal{M}'_\mathcal{A}$ such that $val_{\mathcal{M}'_\mathcal{A}}(g) \geq v$. There exists a memoryless strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_\mathcal{A},f}|w \models \psi] \geq v$. Moreover, f can be computed from g in polynomial time.*

▶ **Lemma 17.** *Let $v \in \mathbb{R}$, and consider a strategy f in $\mathcal{M}_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}_\mathcal{A},f}|w \models \psi] \geq v$. There exists a strategy g in $\mathcal{M}'_\mathcal{A}$ such that $\mathbb{E}[R_{\mathcal{M}'_\mathcal{A},g}] \geq v$.*

Finally, using Theorem 15, and the fact that $\mathcal{A}_\psi$ is doubly exponential in ψ, we can use the same reasoning as in the proof of Theorem 7 and conclude with the following. The proof can be found in the full version.

▶ **Theorem 18.** *Solving the SHQSyn problem with environment assumptions can be done in doubly-exponential time. The corresponding decision problem is 2EXPTIME-complete.*

7 Extensions

In this section we describe two extensions to the setting. The first combines the threshold and assumption extensions presented in Sections 5 and 6. The second shows how to handle a non-uniform probability distribution.

7.1 Combining an Almost-Sure Threshold with Environment Assumptions

Combining an almost-sure threshold with environment assumptions requires some subtlety in the definitions. As an input for the problem, we are given an LTL[$\mathcal{F}$] formula φ over $I \cup O$, an LTL environment assumption ψ over I such that $\Pr(\psi) > 0$, and a threshold $t \in [0, 1]$. Then, we seek a a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi]$ and for which $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t|w \models \psi) = 1$. In particular, the threshold t should be attained almost surely only in computations that satisfy ψ.

▶ Remark 19. Note that it could have also been possible to seek a transducer $\mathcal{T}$ that maximizes $\mathbb{E}[X_{\mathcal{T},\varphi}|w \models \psi]$ and for which $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t) = 1$, namely for which the threshold should hold almost surely regardless of the assumption. We found this approach less appealing. Its solution, however, is a straightforward combination of our constructions. That is, we start with the product $\mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\geq t} \times \mathcal{A}_\psi$, as defined in Sections 5 and 6, apply the reset modification described in Section 6, and seek a maximizing strategy in the resulting MDP. ◀

We solve the problem as follows. We start by checking whether there exists a transducer $\mathcal{T}$ such that $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq v | w \models \psi) = 1$, using the following lemma (see the full version for the proof).

▶ **Lemma 20.** *Let φ, ψ, and t be as above. For every transducer $\mathcal{T}$ it holds that* $\Pr([\![\mathcal{T}(w), \varphi]\!] \geq t | w \models \psi) = 1$ *iff* $\Pr([\![\mathcal{T}(w), \psi \rightarrow \varphi]\!] \geq t) = 1$.

Using Lemma 20, we can decide the existence of a transducer $\mathcal{T}$ as we seek, by constructing the DPW $\mathcal{A}_{\psi \rightarrow \varphi, \geq t}$ as per Theorem 3, and keeping only almost-sure winning states as done in Section 5.

We now proceed as in the first approach, by constructing the product $\mathcal{A}_\ell \times \ldots \times \mathcal{A}_n \times \mathcal{A}'_{\psi \rightarrow \varphi, \geq t} \times \mathcal{A}_\psi$, where $\mathcal{A}'_{\psi \rightarrow \varphi, \geq t}$ is obtained from $\mathcal{A}_{\psi \rightarrow \varphi, \geq t}$ by keeping only almost-sure winning states.

7.2 Handling a Non-Uniform Distribution

In order to handle a non-uniform distribution on the input signals, we first have to decide how to model arbitrary distributions on $(2^I)^\omega$. The common way to do so is to assume that the distribution is generated by a pre-MDP $\mathcal{D} = \langle S, s_0, 2^O, \mathrm{P} \rangle$ and a labeling function $\iota : S \rightarrow 2^I$, where a state $s \in S$ generates the input $\iota(s)$. Thus, the probability of an input signal to hold depends on the history of the interaction with the system. Formally, every run $r = s_0, s_1, ...$ of $\mathcal{D}$ generates an input sequence $\iota(s_1), \iota(s_2)$, and the distribution on runs induces a distribution on $(2^I)^\omega$. [2]

All our results can be adapted to handle a distribution given by $\mathcal{D}$ as above. We only have to change the construction of the achievability MDP described in Section 3.1 as follows. For a pre-automaton $\mathcal{B} = \langle 2^{I \cup O}, Q, q_0, \delta \rangle$ and a distribution pre-MDP $\mathcal{D} = \langle S, s_0, 2^O, \mathrm{P} \rangle$ with labeling function ι, we define the induced pre-MDP as $\mathcal{M}^{\mathcal{D}}_{\mathcal{B}} = \langle Q \times S, \langle q_0, s_0 \rangle, 2^O, \mathrm{P}' \rangle$ where for every two states $\langle q, s \rangle, \langle q', s' \rangle \in Q \times S$ and action $o \in 2^O$, we have $\mathrm{P}'(\langle q, s \rangle, o, \langle q', s' \rangle) = \mathrm{P}(s, o, s')$ if $\delta(q, \iota(s') \cup o) = q'$, and 0 otherwise. It is not hard to see that all the constructions we apply to achievability MDP $\mathcal{M}_{\mathcal{A}}$ can be applied to $\mathcal{M}^{\mathcal{D}}_{\mathcal{A}}$, which would take the distribution in $\mathcal{D}$ into account. The complexity of the algorithms is polynomial in $\mathcal{M}^{\mathcal{D}}_{\mathcal{A}}$. Thus, the complexity of our algorithms remains 2EXPTIME-complete in φ and polynomial in $\mathcal{D}$.

References

1. S. Almagor, O. Kupferman, and Y. Velner. Minimizing expected cost under hard boolean constraints, with applications to quantitative synthesis. In *27th CONCUR*, 2016.
2. C. Baier, J. Klein, S. Klüppelholz, and S. Märcker. Computing conditional probabilities in markovian models efficiently. In *20th TACAS*, pages 515–530, 2014.
3. R. Bloem, K. Chatterjee, T. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *21st CAV*, volume 5643 of *LNCS*, pages 140–156, 2009.
4. R. Bloem, R. Ehlers, and R. Könighofer. Cooperative reactive synthesis. In *13th ATVA*, pages 394–410, 2015.
5. V. Bruyère, E. Filiot, M. Randour, and J-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *31st STACS*, volume 25 of *LIPIcs*, pages 199–213, 2014.
6. K. Chatterjee and L. Doyen. Energy and mean-payoff parity markov decision processes. In *36th MFCS*, pages 206–218, 2011.

[2] Note that we do not consider the label on s_0, in order to allow a distribution on the initial letters.

7 K. Chatterjee, T. Henzinger, and B. Jobstmann. Environment assumptions for synthesis. In *19th CONCUR*, volume 5201 of *LNCS*, pages 147–161, 2008.

8 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Games with secure equilibria. In *19th LICS*, pages 160–169, 2004.

9 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Quantitative stochastic parity games. In *SODA 04*, pages 121–130, 2004.

10 K. Chatterjee, Z. Komárková, and J. Kretínský. Unifying two views on multiple mean-payoff objectives in markov decision processes. In *30th LICS*, pages 244–256, 2015.

11 A. Church. Logic, arithmetics, and automata. In *Proc. Int. Congress of Mathematicians, 1962*, pages 23–35. Institut Mittag-Leffler, 1963.

12 L. Clemente and J-F. Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *30th LICS*, pages 257–268, 2015.

13 M. Faella, A. Legay, and M. Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.

14 J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

15 D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *16th TACAS*, volume 6015 of *LNCS*, pages 190–204, 2010.

16 O. Kupferman, Y. Lustig, M.Y. Vardi, and M. Yannakakis. Temporal synthesis for bounded systems and environments. In *28th STACS*, pages 615–626, 2011.

17 M. Kwiatkowska and D. Parker. Automated verification and strategy synthesis for probabilistic systems. In *11th ATVA*, volume 8172 of *LNCS*, pages 5–22, 2013.

18 W. Li, L. Dworkin, and S. A. Seshia. Mining assumptions for synthesis. In *9th MEMOCODE*, pages 43–50, 2011.

19 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *16th POPL*, pages 179–190, 1989.

20 U. Boker S. Almagor and and O. Kupferman. Formalizing and reasoning about quality. *J. ACM*, 63(3), 2016.

21 S. Schewe and B. Finkbeiner. Bounded synthesis. In *5th ATVA*, volume 4762 of *LNCS*, pages 474–488, 2007.

Hedging Bets in Markov Decision Processes*

Rajeev Alur[1], Marco Faella[2], Sampath Kannan[3], and Nimit Singhania[4]

1 **University of Pennsylvania, Philadelphia, USA**
2 **Università di Napoli "Federico II", Naples, Italy**
3 **University of Pennsylvania, Philadelphia, USA**
4 **University of Pennsylvania, Philadelphia, USA**

Abstract

The classical model of Markov decision processes with costs or rewards, while widely used to formalize optimal decision making, cannot capture scenarios where there are multiple objectives for the agent during the system evolution, but only one of these objectives gets actualized upon termination. We introduce the model of *Markov decision processes with alternative objectives* (MDPAO) for formalizing optimization in such scenarios. To compute the strategy to optimize the expected cost/reward upon termination, we need to figure out how to balance the values of the alternative objectives. This requires analysis of the underlying infinite-state process that tracks the accumulated values of all the objectives. While the decidability of the problem of computing the exact optimal strategy for the general model remains open, we present the following results. First, for a Markov chain with alternative objectives, the optimal expected cost/reward can be computed in polynomial-time. Second, for a single-state process with two actions and multiple objectives we show how to compute the optimal decision strategy. Third, for a process with only two alternative objectives, we present a reduction to the minimum expected accumulated reward problem for one-counter MDPs, and this leads to decidability for this case under some technical restrictions. Finally, we show that optimal cost/reward can be approximated up to a constant additive factor for the general problem.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases Markov decision processes, Infinite state systems, Multi-objective optimization

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.29

1 Introduction

The mathematical model of *Markov decision processes* (MDP) is suitable for modeling decision making in situations where the evolution of a system is partly probabilistic and partly controlled by strategic choices. To define the notion of an *optimal strategy* we need to associate costs (or equivalently, rewards) with an execution of an MDP. Traditionally, this is done by associating a numerical value with each action in each state, and the cost of a terminating execution is the sum of the costs of all the decisions made along the way. The optimization problem then is to minimize the expected cost (or equivalently, maximize the expected reward) over the set of all strategies. It is well known that there exists a memoryless optimal strategy, where-in the globally optimal decision at any step during an execution is a function of the current state, and the optimization problem can be solved in polynomial-time using linear programming [3].

* This research was partially supported by NSF Expeditions award CCF 1138996.

© Rajeev Alur, Marco Faella, Sampath Kannan, and Nimit Singhania;
licensed under Creative Commons License CC-BY

While this classical framework is used in a variety of disciplines such as optimal control, finance, and robotic motion planning, it cannot capture scenarios where there are multiple objectives to optimize but only one of these gets realized in the end. This occurs commonly in real life situations, when there are multiple goals only one of which is achieved in the end. Examples of such scenarios include a candidate applying for jobs, a student applying to schools or a company marketing its product to different audiences. As an illustrative scenario, imagine a venture capitalist (VC) investing in pharmaceutical research, where there are multiple companies all competing to develop a vaccine for malaria. At each point in time, the investor has several choices for how much to invest in each company's research. When one company succeeds in creating the vaccine, it patents the discovery and the investor reaps financial benefits proportional to the total amount (s)he has invested in that company, but does not get any reward for investments in the other companies. We can model the evolution of the system as an MDP. Each state of the system corresponds to the current conditions of all the companies. In each state, each company has a probability of succeeding resulting in termination, or the system continues to evolve probabilistically, partially influenced by the VC's investment choice.

The optimal investment strategy is not immediately obvious and is contingent on the company that succeeds in creating the vaccine. Particularly, it depends on the dynamics of the competition and how investment influences each company. If the competition is positive where investing in one firm boosts the other companies to improve their research, then spreading the investment to each firm is optimal. Whereas, if the competition is negative and investing in one firm demotivates others, then investing in a single best firm is optimal. Note that an objective here is to improve the research of a company and depending on the dynamics, the VC might decide how to balance optimization of every objective.

This scenario can be represented in our formal model of MDPs with alternative objectives, where the MDP is augmented with a set of registers corresponding to the different objectives. At each step, based on the current state and the chosen action, each register is updated by a specified integer amount. Then, following action-dependent transition probabilities, the process either continues in another state or terminates. When terminating, the value of one of the registers is probabilistically chosen as the cost/reward of the whole path. The optimization problem then is to minimize the expected cost or equivalently, maximize the expected reward upon termination over the set of all strategies. In this work, we focus on costs and minimization of the expected cost.

It turns out that for an MDP with alternative objectives, the optimal decision at any step depends not only on the current state but also on the register values. Since the space of register values is unbounded, analysis is challenging. For a Markov chain (that is, an MDP with a single action), we show that the expected cost in a given state is a *linear* function of the register values, whose coefficients can be computed in polynomial-time by solving a system of linear equations (see Section 4).

For an MDP, the optimal expected cost is no longer a linear function of the register values. To solve this general case, in Section 7, we present an *approximation* algorithm that can approximate the optimal expected cost with a specified error ϵ in pseudo-polynomial time, that is, polynomial in the number of states, actions, and the binary encoding of probabilities and ϵ, but exponential in the number of registers and binary encoding of register updates.

We present exact solutions for two special cases: systems with a single state and two actions (Section 5), and two-register systems with an arbitrary number of states and actions, subject to a mild condition (Section 6).

For systems with a single state and two actions, we observe that the choice of the optimal action depends on a *linear* function P of register values with one action being optimal when

P is positive and the other action when P is negative or zero. Moreover, only a finite number of distinct values for P needs to be taken into account, so that the minimal cost problem can be solved by a reduction to a Markov chain with alternative objectives.

When instead the input system has only two registers, we show that optimal strategies only need to track the difference between register values, allowing us to relate our model to one-counter MDPs of Bradzil et al. [6, 5]. However, this is not sufficient to achieve decidability, as the corresponding problem for one-counter MDPs has not been solved either. If the input system is additionally *tie-less* (as defined in Section 6), optimal strategies issue the same action whenever the difference between the registers is greater, in absolute value, than a certain threshold. Thanks to this property, we can reduce our problem to a bounded number of expected accumulated reward problems for probabilistic one-counter automata, which are analogous to one-counter MDPs with a single action. Since the latter problem is decidable, we obtain decidability of our original question.

The exact solution for the general case of MDPs with alternative objectives, even establishing decidability, remains an open problem.

Related work

Optimization problems for MDPs with different cost criteria have been studied extensively (see [10] for an overview and [11] for applications to computer-aided verification), but we are not aware of existing results directly relevant to the model we propose. A closely related model is that of MDPs with multiple objectives (see [12] for an overview), where multiple objectives similar to our model are associated with the MDP. These objectives are mixed together via a *scalarization* function to get the final expected cost/reward, which does not correspond to choosing a single cost/reward based on the final state that we want to model.

Our model with alternative costs is a special case of *cost register automata* that associate numerical costs with strings [1], and for such automata, optimization in a two-player game (without probabilistic transitions) is solvable in PSPACE [2]. Optimization problems for MDPs where the state-space is augmented with an unbounded counter have been studied recently [6, 5, 8]. The reduction of the special case of our model considered in Section 6 has similar state-space, but is different since the cost upon termination is proportional to the counter value.

2 Model

We describe here the model of Markov decision processes with alternative objectives (MDPAO). As in a traditional MDP, the process consists of a set of states and a set of actions. At a state, an action is chosen and based on the action, the process probabilistically transitions to the next state or terminates. Upon termination, the cost of the run of the process is calculated and this is where our model differs from the standard MDP. The process maintains a set of registers that start off at some initial values and are updated at every step depending on the action chosen at the current state. For each register and each action, the update consists of the addition of some integer, possibly negative, to the register value and upon termination, the value of one of the registers is probabilistically chosen as the cost. Given an initial state and the initial values for registers, we consider the problem of computing the minimum expected cost of a run of the process for the optimal choice of actions at each state.

We explain this further using an MDPAO with a single state q, two actions α and β and two registers x and y as shown in Figure 1. On choosing action α (β), with probability 0.5, the process returns back to q and increments x (y) by 1, and with probability 0.5, it

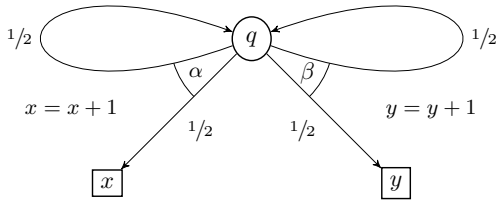


Figure 1 Example MDPAO with two registers x and y.

terminates with value of register x (y) as the cost. A possible path when the process is started at q with $(x, y) = (0, 0)$ is $\rho = (q, (0, 0)) \xrightarrow{\alpha} (q, (1, 0)) \xrightarrow{\beta} (q, (1, 1)) \xrightarrow{\alpha} x$, i.e., the process returns to q with $(x, y) = (1, 0)$ on choosing α, and then with $(x, y) = (1, 1)$ on choosing β and finally, it terminates with value of register $x = 1$ as the cost, on choosing α.

Now we formally define the model. An MDPAO $\mathcal{M}$ consists of the following:

- Q, a finite set of states.
- X, a finite set of registers with $\mathcal{V}$, the set of valuation functions $X \to \mathbb{Z}$ that map registers to their values.
- Γ, a finite set of actions.
- $\delta : Q \times \Gamma \times X \to \mathbb{Z}$, a function that defines the updates to the register values on a state transition. Note that we use $\delta(q, \alpha)$ to refer to a function that maps registers to their updates on action α at state q.
- $p : Q \times \Gamma \times (Q \cup X) \to [0, 1]$, a function that defines the probability of transition to a state or termination with a register value, given a state and the action selected at the state. The probability of transition from q to a state q' when action α is chosen is given by $p(q, \alpha, q')$ and the probability that the process terminates with value of register x as the cost is given by $p(q, \alpha, x)$. Note that, probabilities of transitions out of a state sum to 1, i.e., $\sum_{q' \in Q} p(q, \alpha, q') + \sum_{x \in X} p(q, \alpha, x) = 1$. We assume that $\sum_{x \in X} p(q, \alpha, x) > 0$ for all $q \in Q$ and $\alpha \in \Gamma$ to ensure that probability that a process does not terminate decreases exponentially with the number of steps.

Strategy. A *(full) strategy* is a function $\sigma : Q^+ \times \mathcal{V} \to \Gamma$, that given a sequence of states and initial valuation of registers determines the next action to be chosen at the current state. A *path-oblivious* strategy is a strategy that only depends on the current state and the current value of the registers and hence is a function of type $\sigma : Q \times \mathcal{V} \to \Gamma$. Given a strategy, the choice of action at each state is fixed and the Markov decision process is transformed into a Markov chain.

Path. A *(finite) path* $\rho = (q_0, \nu_0) \xrightarrow{\alpha_0} (q_1, \nu_1) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} (q_n, \nu_n) \xrightarrow{\alpha_n} x, q_i \in Q, \alpha_i \in \Gamma, \nu_i \in \mathcal{V}, x \in X$, is a sequence of states, register values and actions chosen such that, for every transition $(q_i, \nu_i) \xrightarrow{\alpha_i} (q_{i+1}, \nu_{i+1})$, its probability is greater than 0, i.e., $p(q_i, \alpha_i, q_{i+1}) > 0$ and $\nu_{i+1} = \nu_i + \delta(q_i, \alpha_i)$, and the path terminates with value of register x with a probability greater than 0, i.e., $p(q_n, \alpha_n, x) > 0$. A path ρ is *consistent* with a strategy σ if every action α_i chosen in ρ is consistent with σ, i.e., $\alpha_i = \sigma(q_0 q_1 \dots q_i, \nu_0)$. The probability of the process following the path ρ is given by $Pr(\rho) = (\prod_{i=0}^{n-1} p(q_i, \alpha_i, q_{i+1}))p(q_n, \alpha_n, x)$. The cost of the path ρ is defined as $f(\rho) = \nu_n(x)$.

Cost. Let $\Pi_\sigma(q,\nu)$ be the set of all paths that start in $q \in Q$ with register values ν and are consistent with σ, i.e., $\Pi_\sigma(q,\nu) = \{\rho | (q_0,\nu_0) = (q,\nu), \rho \text{ is consistent with } \sigma\}$. Given an initial valuation of registers ν and a strategy σ, the expected cost of a run started from a state q is $f_\sigma(q,\nu) = \sum_{\rho \in \Pi_\sigma(q,\nu)} Pr(\rho) \cdot f(\rho)$.

In the example shown in Figure 1, suppose the process starts with $\nu(x) = 0$ and $\nu(y) = 0$. For a strategy that always chooses α i.e. $\sigma(q^i,\nu) = \alpha$, the process terminates after k steps with probability 0.5^k and cost $k-1$ and thus, the expected cost is $(0\times0.5+1\times0.5^2+2\times0.5^3+\dots) = 1$. Similarly for a strategy that always chooses β, the expected cost is 1. Now for a strategy that alternately chooses α and β, i.e., $\sigma(q^{2i},\nu) = \alpha$ and $\sigma(q^{2i+1},\nu) = \beta$, we get a lower expected cost of $\frac{1}{3}$. In fact, this is the minimum expected cost. Note that, unlike traditional MDPs, the optimal strategy chooses multiple actions at state q.

Problem. Given an MDPAO $\mathcal{M}$, we consider here the problem of computing the minimum expected cost of a run of $\mathcal{M}$ started in an initial state $q \in Q$ with initial valuation of registers $\nu \in \mathcal{V}$. Let $f(q,\nu)$ denote this minimum expected cost and σ^* the optimal strategy, i.e., $f(q,\nu) = \min_\sigma f_\sigma(q,\nu), \sigma^* = \text{argmin}_\sigma f_\sigma(q,\nu)$.

Variations of the model. The above model can be further extended as follows. A first extension is one where the update function δ depends also on the next state, along with the current state and the action chosen. A second extension is one where, upon termination, the cost is a non-negative linear combination of register values rather than a single register's value. Note that our algorithms generalize to these extensions immediately and to keep the presentation simple, we describe solutions only for the model defined above.

3 Useful results

We describe here some results that are useful in solving the problem of computing the minimum expected cost.

We first show that $|f_\sigma(q,\nu)|$ is bounded by a linear function of $\max_{x\in X} |\nu(x)|$ for all strategies σ and thus, $|f(q,\nu)|$ is bounded by this function. We use this result to compute an approximation for the minimum expected cost in Section 7 and to prove subsequent results in this section. Let p_M be the maximum probability of continuation and δ_M be magnitude of maximum change made to any register, in one step of the process. We state the result in the following lemma.

▶ **Lemma 1.** *Given $\nu \in \mathcal{V}$, for all $q \in Q$ and strategies σ,*

$$|f_\sigma(q,\nu)| \leq \left(\frac{\max_{x\in X} |\nu(x)|}{1-p_M} + \frac{p_M \delta_M}{(1-p_M)^2} \right) \triangleq B(\max_{x\in X} |\nu(x)|),$$

where $p_M = \max_{q\in Q,\alpha\in\Gamma} \left(\sum_{q'\in Q} p(q,\alpha,q') \right)$ and $\delta_M = \max_{q\in Q,\alpha\in\Gamma,x\in X} |\delta(q,\alpha,x)|$.

Proof. We claim that for any strategy σ and for all $i \geq 0$, the probability that the process does not terminate in first i steps, p_i is bounded by p_M^i, the absolute value of any register after i steps, ν_i is bounded by $(\max_{x\in X} |\nu(x)| + i\delta_M)$ and thus, the absolute value of the expected cost paid in the $(i+1)$th step by the process, c_i, is bounded by $p_M^i(\max_{x\in X} |\nu(x)| + i\delta_M)$.

We prove this by induction. It is trivially true when $i = 0$. Assuming it is true in the ith step, we prove that it is also true in $(i+1)$th step. Probability that the process continues to the next step at the $(i+1)$th step at any state and for any action chosen is less than p_M and thus, $p_{i+1} \leq p_i p_M = p_M^{i+1}$. Similarly, the register values can be changed by at most δ_M

in a step and thus, the absolute value of registers after $(i+1)$ steps must be bounded by $\max_{x\in X}|\nu(x)| + (i+1)\delta_M$. Note that the absolute value of expected cost paid at a state q with register values ν' for an action α is

$$\big|\sum_{x\in\Gamma} p(q,\alpha,x)\nu'(x)\big| \;\le\; \big(\max_{x\in X}|\nu'(x)|\big)\sum_{x\in\Gamma} p(q,\alpha,x) \;\le\; \max_{x\in X}|\nu'(x)|.$$

Therefore, c_{i+1} is bounded by the maximum probability to reach the $(i+2)$th step $\times$ the expected cost paid in the $(i+2)$th step $\le p_M^{i+1}(\max_{x\in X}|\nu(x)| + (i+1)\delta_M)$.

Now, the absolute value of the total expected cost of strategy σ is less than $\sum_{i=0}^{\infty} c_i \le \sum_{i=0}^{\infty}\big(p_M^i(\max_{x\in X}|\nu(x)| + i\delta_M)\big)$. Since the probability of termination in each step is strictly greater than 0, the maximum probability $p_M < 1$ and the required result follows. ◀

Next, minimum expected cost $f(q,\nu)$ can be expressed recursively using the costs at the next step, $f(q',\nu+\delta(q,\alpha))$. Let a cost function be a function $Q\times\mathcal{V}\to\mathbb{R}$ that maps a state and valuation of registers to a real valued cost and let the set of cost functions be $\mathcal{C}$. We define an operator $T:\mathcal{C}\to\mathcal{C}$ as follows.

$$Tg(q,\nu) = \min_{\alpha\in\Gamma}\Big(\sum_{x\in X} p(q,\alpha,x)\nu(x) + \sum_{q'\in Q} p(q,\alpha,q')g(q',\nu+\delta(q,\alpha))\Big)$$

The minimum expected cost function f is a fixed point of this operator, i.e., $Tf = f$. In fact, we can show that a strategy σ is an optimal strategy if f_σ is a fixed point of T, as stated in Lemma 2. Our proof for Lemma 2 closely follows the proof by Bertsekas et al in [4] to show that the recursive equation describing the stochastic shortest path problem has a unique solution which represents the optimal cost vector. We use this lemma to compute the optimal strategies in Sections 5 and 6. Another consequence of this lemma is that we can limit our investigation to path-oblivious strategies, i.e., strategies that only depend on the current state and the current value of the registers.

▶ **Lemma 2.** *Given a strategy σ in an MDPAO $\mathcal{M}$, the cost function f_σ is a fixed point of T, i.e., $Tf_\sigma(q,\nu) = f_\sigma(q,\nu)$ for all $q\in Q$ and $\nu\in\mathcal{V}$, if and only if σ is an optimal strategy and f_σ is the minimum expected cost function.*

Proof. To prove this lemma, we show that set of the cost functions, realizable by strategies, forms a partially ordered set such that the operator T has a unique fixed point in this set. Since the minimum expected cost function must be the least fixed point of T, any fixed point of T is the required minimum expected cost function.

For a strategy σ, we define a new operator on cost functions, $T_\sigma:\mathcal{C}\to\mathcal{C}$, where $T_\sigma g$ computes the cost of using strategy σ for one step and then paying the cost as given by g. Let $\alpha = \sigma(q,\nu)$, we have:

$$T_\sigma g(q,\nu) = \sum_{x\in X} p(q,\alpha,x)\nu(x) + \sum_{q'\in Q} p(q,\alpha,q')g(q',\nu+\delta(q,\alpha))$$

We also define a complete partial order L on cost functions as follows. Let $f_\top, f_\bot \in \mathcal{C}$ be the two cost functions defined by $f_\top(q,\nu) = B(\max_{x\in X}|\nu(x)|)$ and $f_\bot(q,\nu) = -B(\max_{x\in X}|\nu(x)|)$, where B is defined in Lemma 1. Further, $g\le h$, where $g,h\in\mathcal{C}$, if for all $q\in Q,\nu\in\mathcal{V}$, $g(q,\nu)\le h(q,\nu)$. Now, L is a complete partial order for relation $\le$ on the set $\{g\in\mathcal{C} \mid f_\bot\le g\le f_\top\}$. Note that by Lemma 1, for all strategies σ, $f_\bot\le f_\sigma\le f_\top$ and hence $f_\sigma\in L$.

The operators T and T_σ are closed in L, i.e. $g\in L \Rightarrow Tg, T_\sigma g\in L$ since $Tf_\top \le f_\top, T_\sigma f_\top \le f_\top$ and $Tf_\bot \ge f_\bot, T_\sigma f_\bot \ge f_\bot$ (by Lemma 1). Further, operators T and T_σ

are monotonic and continuous, since they apply linear transformations and the minimum operator on cost functions, which preserve both properties. Hence, by Kleene's fixed point theorem, both T and T_σ must have a least fixed point in L given by $\lim_{k\to\infty} T^k f_\perp$ and $\lim_{k\to\infty} T_\sigma^k f_\perp$ respectively.

Next, we prove that T_σ has a unique fixed point in L. To prove this, we show that for all $g, h \in L$, $\lim_{k\to\infty} T_\sigma^k g(q,\nu) = \lim_{k\to\infty} T_\sigma^k h(q,\nu)$. Note that, $T_\sigma^k g$ corresponds to the expected cost of using strategy σ for first k steps and then terminating with the cost given by g at the $(k+1)$th step. Hence, the difference $T_\sigma^k g(q,\nu) - T_\sigma^k h(q,\nu)$ corresponds to the difference in costs paid at the $(k+1)$th step, since the cost paid till k steps is same for both $T_\sigma^k g(q,\nu)$ and $T_\sigma^k h(q,\nu)$. As described in the proof for Lemma 1, the probability of the process not terminating in first k steps is bounded by p_M^k and the magnitude of register values are bounded by $(\max_{x\in X} |\nu(x)| + k\delta_M)$. Since $g, h \in L$, we have:

$$|T_\sigma^k g(q,\nu) - T_\sigma^k h(q,\nu)| \leq 2p_M^k B\big(\max_{x\in X} |\nu(x)| + k\delta_M\big).$$

Note that p_M^k decreases exponentially with k, whereas $B(\max_{x\in X} |\nu(x)| + k\delta_M)$ increases only linearly with k. Hence, $\lim_{k\to\infty}(T_\sigma^k g(q,\nu) - T_\sigma^k h(q,\nu)) = 0$, and thus T_σ has a unique fixed point in L.

Now we prove that T has a unique fixed point in L. Suppose T had two fixed points in L, say f_1, f_2. Then we can find strategies σ_1, σ_2 such that $T_{\sigma_1} f_1 = f_1$ and $T_{\sigma_2} f_2 = f_2$ by using the actions that minimize Tf_1 and Tf_2 respectively. Further, $Tf_1 \leq T_{\sigma_2} f_1$ since Tf_1 corresponds to the minimum of the costs for different actions, while $T_{\sigma_2} f_1$ corresponds to one of these costs. This implies $f_1 = \lim_{k\to\infty} T^k f_1 \leq \lim_{k\to\infty} T_{\sigma_2}^k f_1 = f_2$ and thus, $f_1 \leq f_2$. By a symmetric argument, $f_2 \leq f_1$. Hence, T must have a unique fixed point in L. The minimum expected cost f is the least fixed point of T in L and hence, any fixed point of T in L is the required minimum expected cost. ◀

Our final result shows that the optimal strategy σ^* depends only on the relative difference between the initial register values and not their absolute values. We state it in Lemma 3. We prove this by showing that $f(q,\nu+k) - k$ is also a fixed point of T and hence, must be equal to $f(q,\nu)$. Note that, this result can be used to reduce a model with $|X|$ registers into a model with $|X|-1$ registers. We use it to simplify the problem in Section 6.

▶ **Lemma 3.** *Given an MDPAO $\mathcal{M}$, for all $q \in Q, \nu \in \mathcal{V}, k \in \mathbb{Z}$, it holds that $f(q,\nu+k) = f(q,\nu) + k$, where $(\nu+k)(x) = \nu(x) + k$ for all $x \in X$.*

Proof. We use Lemma 2 to prove this. Let $g \in \mathcal{C}$ be a function such that $g(q,\nu) = f(q,\nu+k) - k$ for all $q \in Q$ and $\nu \in \mathcal{V}$. Note that g may not lie in the complete partial order L as described in the proof for Lemma 2. However if we define the partial order L' using $f'_\top = f_\top + |k|(1 + \frac{1}{1-p_M})$ and $f'_\perp = f_\perp - |k|(1 + \frac{1}{1-p_M})$, the proof of Lemma 2 still follows and T must have a unique fixed point in L'. Also g belongs to L' since $|g(q,\nu)| \leq |f(q,\nu+k)| + |k|$. Further,

$$\begin{aligned} Tg(q,\nu) &= T(f(q,\nu+k) - k) \\ &= Tf(q,\nu+k) - k \\ &= f(q,\nu+k) - k \\ &= g(q,\nu). \end{aligned}$$

Hence, g is a fixed point of T and thus, by Lemma 2, it must be the minimum expected cost function f. ◀

4 Markov chains with alternative objectives

We consider a special case of the problem when the set of actions consists of a single action, i.e. $\Gamma = \{\alpha\}$. Since f is a fixed point for the operator T as defined in Section 3, for all $q \in Q$ and $\nu \in \mathcal{V}$,

$$f(q,\nu) = \sum_{x \in X} p(q,x)\nu(x) + \sum_{q' \in Q} p(q,q')f(q',\nu + \delta(q)). \tag{1}$$

The above system of equations may have more than one solutions. However, we show that $f(q,\nu)$ is a linear function of the initial register values ν for all $q \in Q$, as stated in Lemma 4. The cost incurred by a path ρ consists of two components: the initial value of a register x, $\nu(x)$ and the updates to x along the path which are independent of ν. Also, the contribution of $\nu(x)$ to the expected cost $f(q,\nu)$ depends only on the probability with which paths starting at (q,ν) end in x, which again is independent of ν. Therefore, $f(q,\nu)$ must be linear in ν.

▶ **Lemma 4.** *Given an MDPAO with a single action, the minimum expected cost $f(q,\nu)$ is linear in ν, i.e., for all $q \in Q$ and $\nu \in \mathcal{V}$, it holds that $f(q,\nu) = \sum_{x \in X} a_{q,x}\nu(x) + f(q,\nu_0)$, where $\nu_0(x) = 0$ for all $x \in X$ and $a_{q,x}$ is the probability that a path starting at (q,ν) ends in register x.*

Proof. To prove the lemma we show that $f(q,\nu + \delta_x) = f(q,\nu) + a_{q,x}d$, where $\delta_x(x) = d$ and $\delta_x(y) = 0$ for all $y \in X \setminus \{x\}$, for some $d \in \mathbb{Z}$. Let $\rho = (q_0,\nu_0) \xrightarrow{\alpha_0} (q_1,\nu_1) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} (q_n,\nu_n) \xrightarrow{\alpha_n} y$ and $\rho' = (q_0,\nu'_0) \xrightarrow{\alpha_0} (q_1,\nu'_1) \xrightarrow{\alpha_1} \dots \xrightarrow{\alpha_{n-1}} (q_n,\nu'_n) \xrightarrow{\alpha_n} y$ be paths that follow the same sequence of states but are started with different valuations of registers ν and $\nu + \delta_x$ respectively. We can see that $\nu'_i = \nu_i + \delta_x$ for all i, since the same changes are made to both ν and ν' along the way. Therefore, if a path ρ' terminates in x, denoted by $\rho' \rightsquigarrow x$, then $f(\rho') = \nu'_n(x) = \nu_n(x) + d = f(\rho) + d$, and $f(\rho') = f(\rho)$ otherwise. Also note that $Pr(\rho') = Pr(\rho)$.

By definition, $f(q,\nu + \delta_x) = \sum_{\rho' \in \Pi(q,\nu+\delta_x)}(Pr(\rho')f(\rho'))$. Since $f(\rho') = f(\rho) + d$ for all runs $\rho' \rightsquigarrow x$, $f(q,\nu + \delta_x) = f(q,\nu) + d\sum_{\rho \in \Pi(q,\nu),\rho \rightsquigarrow x} Pr(\rho)$ which implies $f(q,\nu + \delta_x) = f(q,\nu) + a_{q,x}d$. The required lemma follows immediately. ◀

Now, from (1) and Lemma 4, we can compute $f(q,\nu)$ by solving the following system of linear equations.

$$a_{q,x} = p(q,x) + \sum_{q' \in Q} p(q,q')a_{q',x}$$

$$f(q,\nu_0) = \sum_{q' \in Q} p(q,q') \left(\sum_{x \in X} a_{q',x}\delta(q,x) + f(q',\nu_0) \right)$$

The above system consists of $|Q| \times (|X| + 1)$ equations and variables and thus, can be solved in $O(|Q|^3|X|^3)$ time using a linear constraint solver.

▶ **Theorem 5.** *The problem of computing $f(q,\nu)$ for $q \in Q$ and $\nu \in \mathcal{V}$ for an MDPAO $\mathcal{M}$ with a single action can be solved in $O(|Q|^3|X|^3)$ time.*

5 Two action single state MDPs with alternative objectives

In this section, we solve the problem for an MDPAO $\mathcal{M}$ where $Q = \{q\}$ and $\Gamma = \{\alpha,\beta\}$. To simplify notation, let $a_x = p(q,\alpha,x)$, $b_x = p(q,\beta,x)$, $a_0 = p(q,\alpha,q)$ and $b_0 = p(q,\beta,q)$. Further, let the register updates be $\delta_\alpha = \delta(q,\alpha)$ and $\delta_\beta = \delta(q,\beta)$.

We observe that in this case while the minimum expected cost $f(\nu)$ is no longer a linear function of ν, the choice of optimal action in the optimal strategy does depend on a linear preference function P of current register values. So if $P(\nu) \leq 0$ then the optimal action at ν is α i.e. $\sigma^*(\nu) = \alpha$, and otherwise $\sigma^*(\nu) = \beta$.

To define P, we need to consider the change in preference ΔP on taking actions α and β. Since $P(\nu)$ is a linear function of ν, the change in preference $\Delta P(\delta) = P(\nu + \delta) - P(\nu)$ depends only on the change in register values δ. We define $\Delta P(\delta)$ as follows:

$$\Delta P(\delta) = \sum_{x \in X} \left(\frac{a_x}{1 - a_0} - \frac{b_x}{1 - b_0} \right) \delta(x).$$

Now $\Delta P(\delta_\alpha)$ and $\Delta P(\delta_\beta)$ capture the change in preference on taking the two actions, where δ_α and δ_β are the corresponding register updates. Depending on whether these values are positive or negative, we can have four possible scenarios.

To illustrate this further, consider the example shown in Figure 1 which is also an instance of this model. On choosing α, the process either terminates with value of register x or increments x and returns to q. Note that if the process does not terminate, the value of x increases and we are likely to pay a higher cost by choosing α in the next step. Hence, our preference P to choose β increases. Similarly on choosing β, our preference P to choose β decreases. This corresponds to the case where $\Delta P(\delta_\alpha) \geq 0$ and $\Delta P(\delta_\beta) < 0$ and the optimal strategy oscillates between choosing the two actions.

Now we give a concrete definition of the preference function P for the different scenarios described above. Let $f_w(\nu)$ be the cost of an infinite sequence of actions given by an infinite word $w = w_1 w_2 w_3 \ldots$ in $\{\alpha, \beta\}^\omega$. Note that $f_{\alpha w}(\nu) = \sum_{x \in X} a_x \nu(x) + a_0 f_w(\nu + \delta_\alpha)$ and $f_{\beta w}(\nu) = \sum_{x \in X} b_x \nu(x) + b_0 f_w(\nu + \delta_\beta)$. We can also compute $f_{\alpha^\omega}(\nu)$ by reducing $\mathcal{M}$ to a Markov chain where action α is chosen always, and $f_{\alpha^\omega}(\nu) = \sum_{x \in X} \left(\frac{a_x \nu(x)}{1 - a_0} + \frac{a_0 a_x \delta_\alpha(x)}{(1-a_0)^2} \right)$. Using this, we can further compute $f_{\beta\alpha^\omega}(\nu)$. We can compute $f_{\beta^\omega}(\nu)$ and $f_{\alpha\beta^\omega}(\nu)$ similarly. Further, for all infinite words w, the difference $f_{\alpha\beta w}(\nu) - f_{\beta\alpha w}(\nu) = \sum_{x \in X} (((1 - b_0)a_x - (1 - a_0)b_x)\nu(x) + a_0 b_x \delta_\alpha(x) - b_0 a_x \delta_\beta(x))$ and is independent of w. We abbreviate this difference as $f_{\alpha\beta\cdot}(\nu) - f_{\beta\alpha\cdot}(\nu)$. We use these quantities to define $P(\nu)$ and the optimal strategy σ^* in Lemma 6.

▶ **Lemma 6.** *In a two action single state MDPAO $\mathcal{M}$, if $P(\nu) \leq 0$, then the optimal strategy at ν, $\sigma^*(\nu) = \alpha$ and otherwise $\sigma^*(\nu) = \beta$, where $P(\nu)$ is defined as follows:*

1. *If $\Delta P(\delta_\alpha) \geq 0$ and $\Delta P(\delta_\beta) < 0$, $P(\nu) = f_{\alpha\beta\cdot}(\nu) - f_{\beta\alpha\cdot}(\nu)$.*
2. *If $\Delta P(\delta_\alpha) \geq 0$ and $\Delta P(\delta_\beta) \geq 0$, $P(\nu) = f_{\alpha\beta^\omega}(\nu) - f_{\beta^\omega}(\nu)$.*
3. *If $\Delta P(\delta_\alpha) < 0$ and $\Delta P(\delta_\beta) < 0$, $P(\nu) = f_{\alpha^\omega}(\nu) - f_{\beta\alpha^\omega}(\nu)$.*
4. *If $\Delta P(\delta_\alpha) < 0$ and $\Delta P(\delta_\beta) \geq 0$, $P(\nu) = f_{\alpha^\omega}(\nu) - f_{\beta^\omega}(\nu)$.*

Proof. To prove this lemma, we show that in each case, f_{σ^*} is a fixed point of T and then by Lemma 2, σ^* is the optimal strategy. To compute the optimal action at ν, we need to consider the difference between the minimum expected cost on choosing action α and that on choosing β. This would require us to consider all possible sequences of actions starting from α and β and compute the expected cost for each sequence, which would be difficult. However, Lemma 2 helps us break down the problem into recursive cases, and consider only a few of sequences of actions for both α and β and whichever leads to a lower cost gives the desired optimal action. For Cases 1, 2 and 3, we also give alternate proofs, since f_{σ^*} is only recursively defined in these cases and a closed form representation is not available.

Case 4: $\Delta P(\delta_\alpha) < 0, \Delta P(\delta_\beta) \geq 0, P(\nu) = f_{\alpha^\omega}(\nu) - f_{\beta^\omega}(\nu)$. It is easy to see that $f_{\sigma^*}(\nu) = \min(f_{\alpha^\omega}(\nu), f_{\beta^\omega(\nu)})$. This is because, if α is preferred in the current state at ν, it is also preferred in all subsequent steps and thus, α^ω should lead to the minimum expected cost. Similarly, for β. We need to show that $Tf_{\sigma^*}(\nu) = f_{\sigma^*}(\nu)$ for all $\nu \in \mathcal{V}$. Suppose $P(\nu) \leq 0$. Then, $P(\nu + \delta_\alpha) \leq 0$ since $\Delta P(\delta_\alpha) < 0$ and thus, $f_{\sigma^*}(\nu + \delta_\alpha) = f_{\alpha^\omega}(\nu + \delta_\alpha)$. Therefore, the cost of taking action α at ν is $c_\alpha = f_{\alpha^\omega}(\nu)$. If $P(\nu + \delta_\beta) > 0$, then the cost of taking action β, at ν is $c_\beta = f_{\beta^\omega}(\nu)$ and therefore, $Tf_{\sigma^*}(\nu) = \min(c_\alpha, c_\beta) = f_{\sigma^*}(\nu)$. If $P(\nu + \delta_\beta) < 0$, then $c_\beta = f_{\beta\alpha^\omega}(\nu)$. But we can show that $f_{\alpha^\omega}(\nu) - f_{\beta\alpha^\omega}(\nu) = f_{\alpha^\omega}(\nu) - f_{\beta^\omega}(\nu) - (f_{\beta\alpha^\omega}(\nu) - f_{\beta^\omega}(\nu)) = P(\nu) - b_0 P(\nu + \delta_\beta) = (1 - b_0)P(\nu) - b_0 \Delta P(\delta_\beta) \leq 0$. This implies $f_{\alpha^\omega}(\nu) \leq f_{\beta\alpha^\omega}(\nu)$ and therefore, $Tf_{\sigma^*}(\nu) = f_{\sigma^*}(\nu)$. The other case, $P(\nu) > 0$ is symmetric and thus, σ^* is the optimal strategy.

Case 1: $\Delta P(\delta_\alpha) \geq 0, \Delta P(\delta_\beta) < 0, P(\nu) = f_{\alpha\beta.}(\nu) - f_{\beta\alpha.}(\nu)$. Note that the optimal strategy σ^* here oscillates between α and β. Thus, the cost f_{σ^*} can be defined only recursively and its exact representation can not be known a priori. Therefore, we use the recursive definition to show that it is the fixed point. We also give an alternate proof based on an exchange argument for clarity.

We define $f_{\sigma^*}(\nu)$ as follows. If $P(\nu) \leq 0, f_{\sigma^*}(\nu) = f_{\alpha\sigma^*}(\nu) = \sum_{x \in X} a_x \nu(x) + a_0 f_{\sigma^*}(\nu + \delta_\alpha)$ else, $f_{\sigma^*}(\nu) = f_{\beta\sigma^*}(\nu)$. We also assume that $f_{\sigma^*}(\nu)$ is the minimum expected cost i.e. $f_{\sigma^*}(\nu) \leq f_w(\nu)$ for all $w \in \{\alpha, \beta\}^\omega$ and $\nu \in \mathcal{V}$. Now we show that $Tf_{\sigma^*} = f_{\sigma^*}$. Suppose $P(\nu) \leq 0$. Then since $\Delta P(\delta_\beta) < 0, P(\nu + \delta_\beta) < 0$ and $f_{\sigma^*}(\nu + \delta_\beta) = f_{\alpha\sigma^*}(\nu + \delta_\beta)$ or α is the preferred action at $\nu + \delta_\beta$. Thus on taking action β at ν, the cost is $c_\beta = f_{\beta\alpha\sigma^*}(\nu)$. Since $P(\nu) \leq 0$, $f_{\alpha\beta\sigma^*}(\nu) \leq f_{\beta\alpha\sigma^*}(\nu)$ and thus, $f_{\alpha\beta\sigma^*}(\nu) \leq c_\beta$. Further by assumption, $c_\alpha = f_{\alpha\sigma^*}(\nu)$ is the minimum expected cost of taking action α at ν and thus, $c_\alpha \leq f_{\alpha\beta\sigma^*}(\nu) \leq c_\beta$. Hence $Tf_{\sigma^*}(\nu) = \min\{c_\alpha, c_\beta\} = f_{\alpha\sigma^*}(\nu) = f_{\sigma^*}(\nu)$. Further by assumption c_α and c_β are the minimum expected costs of taking actions α and β and thus $Tf_{\sigma^*}(\nu)$ is the minimum expected cost for all ν. The sub-case when $P(\nu) > 0$ is symmetric and thus, σ^* is the optimal strategy.

Alternate proof. We now show a proof based on an exchange argument. Suppose $P(\nu) \leq 0$. We can show that every infinite sequence of actions at ν can be transformed to a sequence starting with action α with a lower expected cost, by exchanging pairs $\beta\alpha$ with $\alpha\beta$. Formally, we show that for all infinite sequence of actions $\beta w,$, there exists w' such that $f_{\alpha w'}(\nu) \leq f_{\beta w}(\nu)$. First, we can show by induction on k that for all $k > 0, z \in \{\alpha, \beta\}^\omega$, $f_{\alpha\beta^k z}(\nu) \leq f_{\beta^k \alpha z}(\nu)$. Base case is implied by $P(\nu) \leq 0$. Assuming it is true for $k - 1$, we show that it holds for k. Since $\Delta P(\delta_\beta) \leq 0$, we can see that $P(\nu + (k-1)\delta_\beta) \leq 0$ and thus, $f_{\beta^{k-1}\alpha\beta z}(\nu) \leq f_{\beta^k \alpha z}(\nu)$. Further, by inductive hypothesis $f_{\alpha\beta^{k-1} z}(\nu) \leq f_{\beta^{k-1}\alpha z}(\nu)$ for all z and thus, $f_{\alpha\beta^k z}(\nu) \leq f_{\beta^k \alpha z}(\nu)$. Further, we can show that $f_{\alpha\beta^\omega}(\nu) \leq f_{\beta^\omega}(\nu)$ because, $f_{\alpha\beta^\omega}(\nu) - f_{\beta^\omega}(\nu) = \sum_{i=0}^{\infty} b_0^i P(\nu + i\delta_\beta) \leq 0$. Hence, for infinite sequence of actions, βw, there is a sequence starting with α that has a lower cost and thus, α must be the optimal action at ν. Similarly, we can show that when $P(\nu) > 0, \sigma^*(\nu) = \beta$.

Case 2: $\Delta P(\delta_\alpha) \geq 0, \Delta P(\delta_\beta) \geq 0, P(\nu) = f_{\alpha\beta^\omega}(\nu) - f_{\beta^\omega}(\nu)$. We again give two proofs, one based on fixed point and other based on an exchange argument like in Case 1.

We define $f_{\sigma^*}(\nu)$ as follows. If $P(\nu) > 0, f_{\sigma^*}(\nu) = f_{\beta^\omega}(\nu)$ else $f_{\sigma^*}(\nu) = f_{\alpha\sigma^*}(\nu)$. Also, like in Case 1, we assume that $f_{\sigma^*}(\nu)$ is the minimum expected cost i.e. $f_{\sigma^*}(\nu) \leq f_w(\nu)$ for all $w \in \{\alpha, \beta\}^\omega$ and $\nu \in \mathcal{V}$. We show that $Tf_{\sigma^*} = f_{\sigma^*}$. Suppose $P(\nu) \geq 0$. Then $P(\nu + \delta_\beta) \geq 0$, $P(\nu + \delta_\alpha) \geq 0$, and $c_\alpha = f_{\alpha\beta^\omega}(\nu)$, $c_\beta = f_{\beta^\omega}(\nu)$. Since $P(\nu) \geq 0$, $c_\alpha \geq c_\beta$

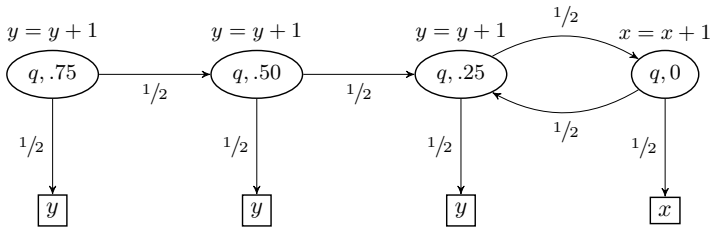


Figure 2 Markov chain conversion of example in Figure 1 when initially $\nu(x) = 8$ and $\nu(y) = 5$ where $f(\nu)$ is the expected cost at $(q, .75)$.

and $Tf_{\sigma^*}(\nu) = f_{\beta^\omega}(\nu) = f_{\sigma^*}(\nu)$. Next, suppose $P(\nu) < 0$. Then, if $P(\nu + \delta_\beta) > 0$, $c_\beta = f_{\beta^\omega}(\nu) > f_{\alpha\beta^\omega}(\nu) \geq c_\alpha$. If $P(\nu + \delta_\beta) \leq 0$, then $c_\beta = f_{\beta\alpha w}(\nu)$ for some $w \in \{\alpha, \beta\}^\omega$. However, $f_{\alpha\beta w}(\nu) - f_{\beta\alpha w}(\nu) = f_{\alpha\beta^\omega}(\nu) - f_{\beta\alpha\beta^\omega}(\nu) = f_{\alpha\beta^\omega}(\nu) - f_{\beta^\omega}(\nu) + f_{\beta^\omega}(\nu) - f_{\beta\alpha\beta^\omega}(\nu) = P(\nu) - b_0 P(\nu + \delta_\beta) = (1 - b_0)P(\nu) - b_0(1 - a_0)\Delta P(\delta_\beta) < 0$. Thus, $c_\beta > f_{\alpha\beta w}(\nu) \geq c_\alpha = f_{\alpha\sigma^*}(\nu)$. Hence, $Tf_{\sigma^*}(\nu) = f_{\sigma^*}(\nu)$ for all ν and therefore σ^* is the optimal strategy.

Alternate proof. We again use an exchange argument to give an alternate proof. First we show that the the optimal strategy at ν is of the form $\alpha^k\beta^\omega$ for some $k \geq 0$. Let $R(\nu) = f_{\alpha\beta w}(\nu) - f_{\beta\alpha w}(\nu)$. Note that $\Delta R(\nu) = (1 - b_0)\Delta P(\nu)$. If $R(\nu) \geq 0$, then by similar argument as in Case 1, $\sigma^*(\nu) = \beta$. In fact, $R(\nu + i\delta_\beta) \geq 0$ and hence, $\sigma^*(\nu + i\delta_\beta) = \beta$ for all $i \geq 0$. Therefore, the optimal sequence of actions at ν is β^ω or $f_{\sigma^*}(\nu) = f_{\beta^\omega}(\nu)$. Suppose $R(\nu) < 0$. We show that for each sequence of actions $w = w_1 w_2 \cdots \in \{\alpha, \beta\}^\omega$, $f_{\alpha^k\beta^\omega}(\nu) \leq f_w(\nu)$ for some k. Let ν_i represent the value of registers after the sequence of actions $w_1 w_2 \ldots w_i$. Let $l = \{i \mid R(\nu_i) \geq 0, R(\nu_j) < 0 \text{ for all } j < i\}$. Thus, l is the first index such that $R(\nu_l) \geq 0$ and for all $i < l, R(\nu_i) < 0$. From the argument above, cost of $w' = w_1 w_2 \ldots w_l \beta^\omega$ is lower than that of w. Further, we can show that cost of $\alpha^k\beta^{l-k}\beta^\omega$ is smaller than w'. Since $R(\nu_j) < 0$ for all $j < l$, every pair $\beta\alpha$ in $w_1 w_2 \ldots w_l$ can be exchanged with $\alpha\beta$ to improve the strategy and thus, give the required result. Thus, the optimal strategy is of the form $\alpha^k\beta^\omega$ for some $k \geq 0$.

If $P(\nu) < 0$, $f_{\alpha\beta^\omega}(\nu) < f_{\beta^\omega}(\nu)$ and hence, α is the optimal action at ν. Further, if $P(\nu) \geq 0$, then for all k, $P(\nu + k\delta_\alpha) \geq 0$, thus, $f_{\beta^\omega}(\nu) \leq f_{\alpha^k\beta^\omega}(\nu)$. Therefore, β is the optimal action at ν and hence, σ^* is the required optimal strategy.

Case 3: $\Delta P(\delta_\alpha) < 0, \Delta P(\delta_\beta) < 0, P(\nu) = f_{\alpha^\omega}(\nu) - f_{\beta\alpha^\omega}(\nu)$. This case is symmetric to Case 2 and can be proven similarly. ◀

Given a two action single state MDPAO $\mathcal{M}$, we compute the minimum expected cost $f(\nu)$ as follows. We first compute the optimal strategy using Lemma 6, then convert the MDP $\mathcal{M}$ into a Markov chain $\mathcal{M}'$ and use $\mathcal{M}'$ to compute the expected cost. In the conversion, since the optimal strategy at a state depends on $P(\nu)$, we associate this value with each state in $\mathcal{M}'$. We start with a state annotated with $P(\nu)$ and, based on the action chosen, transition to a state labelled with $P(\nu + \delta)$, where the registers are incremented by δ. Note that if we reach a state in $\mathcal{M}'$ such that $P(\nu) > 0$ and $\Delta P(\delta_\beta) \geq 0$, then we always choose β after this step, and thus transition back to the same state. Similarly when $P(\nu) \leq 0$ and $\Delta P(\delta_\alpha) < 0$. Figure 2 shows the transformation of the example in Figure 1 into a Markov chain where initially, $\nu(x) = 8$ and $\nu(y) = 5$ and $f(\nu)$ is the expected cost at state $(q, .75)$.

The converted chain consists of a cycle with $\left(\frac{L}{|\Delta P(\delta_\alpha)|} + \frac{L}{|\Delta P(\delta_\beta)|}\right)$ nodes, where L is the least common multiple of $|\Delta P(\delta_\alpha)|$ and $|\Delta P(\delta_\beta)|$. Note that the number of nodes is finite only

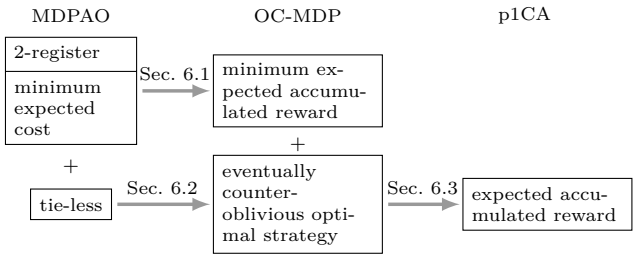


Figure 3 Diagram of the reductions between different models developed in Section 6.

if L is finite. Further, it might consist of a sequence of $O\left(\frac{P(\nu)}{|\Delta P(\delta_\alpha)|}+\frac{P(\nu)}{|\Delta P(\delta_\beta)|}\right)$ states leading to the cycle. Thus, time to compute cost function in the cycle is $O\left(\left(\frac{L}{|\Delta P(\delta_\alpha)|}+\frac{L}{|\Delta P(\delta_\beta)|}\right)^3|X|^3\right)$ and $f(\nu)$ can be computed in $O\left(\left(\frac{P(\nu)}{|\Delta P(\delta_\alpha)|}+\frac{P(\nu)}{|\Delta P(\delta_\beta)|}\right)|X|\right)$ time using the costs in the cycle.

▶ **Theorem 7.** *The problem of computing $f(q,\nu)$ for a two action single state MDPAO $\mathcal{M}$, i.e., $\Gamma=\{\alpha,\beta\}$ and $Q=\{q\}$, can be solved in time*

$$O\Big(\Big(\frac{L}{|\Delta P(\delta_\alpha)|}+\frac{L}{|\Delta P(\delta_\beta)|}\Big)^3|X|^3+\Big(\frac{P(\nu)}{|\Delta P(\delta_\alpha)|}+\frac{P(\nu)}{|\Delta P(\delta_\beta)|}\Big)|X|\Big),$$

where L is the l.c.m. of $|\Delta P(\delta_\alpha)|$ and $|\Delta P(\delta_\beta)|$. Note that if $|\Delta P(\delta_\alpha)|/|\Delta P(\delta_\beta)|$ is irrational, then L is infinite.

6 Two register MDPs with alternative objectives

In this section, we tackle the problem for MDPAOs with two registers x, y, and an arbitrary number of states and actions. Note that by Lemma 3, the optimal strategy depends only on the relative difference between the registers and the strategy needs to maintain a single counter that keeps track of the difference between the values of the two registers.

We show the following results in this section. First, we reduce our problem to the minimum expected accumulated reward problem for one-counter MDPs (OC-MDPs), which is another decision problem whose decidability status is open. Next, we show that for the class of *tie-less* MDPAOs, the optimal strategy is *eventually counter-oblivious*, i.e., there exists a natural number k such that in each state the strategy plays in the same way for all values of the counter higher than k. In turn, this property implies decidability by a reduction of the corresponding OC-MDP to the expected accumulated reward problem for probabilistic 1-counter automata (p1CAs). This sequence of results is summarized in Figure 3.

6.1 Reduction to one-counter Markov decision processes

The problem of computing the minimum expected cost of a 2-register MDPAO can be easily reduced to the minimum expected accumulated reward problem for one-counter Markov decision processes (OC-MDPs) [6, 5]. We employ OC-MDPs *with boundary*, which means that the counter value is always non-negative.

A one-counter MDP is a tuple $\mathcal{A}=(S,\Gamma,\Delta_0,\Delta_{>0})$ [1], where S is a finite set of control states, Γ is a finite set of actions, and Δ_0, $\Delta_{>0}$ are the transitions, with the intended meaning

[1] For technical convenience, our presentation of OC-MDPs includes explicit actions and rewards attached to transitions. It is straightforward to convert this form to the one of Brádzil et al. [6, 8], where rewards are represented by a *simple reward function*.

that Δ_0 applies when the counter is zero and $\Delta_{>0}$ applies otherwise. Each transition is a tuple of the type:

$$\text{(source, action, probability, counter update, reward, destination)}.$$

So, we have $\Delta_0 \subseteq S \times \Gamma \times [0,1] \times \{0,1\} \times \mathbb{Q}^+ \times S$ and $\Delta_{>0} \subseteq S \times \Gamma \times [0,1] \times \{-1,0,1\} \times \mathbb{Q}^+ \times S$. Moreover, for all $s \in S$ and $\gamma \in \Gamma$, the sum of the probabilities of all transitions in Δ_0 (resp., $\Delta_{>0}$) starting from s and γ is 1. A transition $(s, \alpha, p, d, r, s') \in \delta_{>0}$ signifies that when the system is in control state s with a positive value of its counter, action α may lead to state s' with probability p, while updating the counter value from n to $n + d$ and obtaining reward r.

The reward associated with a finite run is the sum of the rewards of each transition. The minimum expected accumulated reward problem takes as inputs an OC-MDP, two control states s, s', and an initial counter value n, and consists of computing the minimum over all policies of the expected reward along all runs that start in (s, n) and end in $(s', 0)$. As usual, by "computing a value" we mean decide whether such value is smaller than or equal to a given rational number.

Consider a 2-register MDPAO $\mathcal{M}$ with non-negative register updates. In order to convert it into an OC-MDP, we first restrict the register updates to the values $\{0, 1\}$. This can be easily accomplished by adding more states and splitting a transition with updates $(+d_x, +d_y)$ into a sequence of $\max\{d_x, d_y\}$ transitions with $\{0, 1\}$ updates. Notice that the newly added transitions violate the assumption that each transition carries a positive probability of termination. However, in the resulting system each *sequence of k transitions* carries a positive probability of termination, for a bounded k. Hence, the only consequence is a slight modification to the bound provided by Lemma 1.

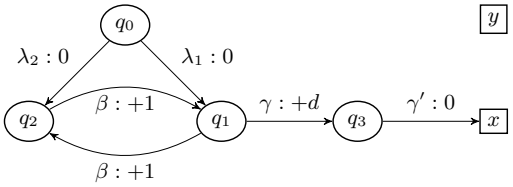
Once register updates have been simplified, the reduction is based on the following idea. Let $f(q, x, y)$ represent the minimum expected cost when the initial state is q and initial values of registers are given by x and y. By Lemma 3, we know that

$$f(q, x, y) = f(q, x - y, 0) + y = f(q, 0, y - x) + x. \tag{2}$$

Clearly, at least one of $x - y$ and $y - x$ is non-negative. By (2), to compute the minimum expected cost of an arbitrary configuration (q, x, y) it is sufficient to know the minimum expected cost of all configurations of the type $(q, z, 0)$ and $(q, 0, z)$ for all $z \geq 0$. All these configurations can be encoded by an OC-MDP, whose counter encodes z and whose control states are pairs (q, r), where $r \in \{x, y\}$ identifies which register has value z. An equivalent way to look at this encoding is that the counter stores the difference between the register that currently holds the highest value and the other register, and the flag r encodes which of the two registers holds the highest value.

It remains to encode the cost structure of the MDPAO. To this purpose, suppose that we are in the configuration $(q, z, 0)$, for some $z \geq 0$ (encoded by the OC-MDP state (q, x) with counter value z), and we want to simulate the effect of an action γ, which leads to state q' with probability $p(q, \gamma, q')$ and carries register updates $\delta(q, \gamma, x) = d_x$ and $\delta(q, \gamma, y) = d_y$. Then, Lemma 3 tells us that $f(q', z + d_x, d_y) = f(q', z + d_x - d_y, 0) + d_y$. So, assuming that $z + d_x - d_y \geq 0$, the corresponding OC-MDP will move with probability $p(q, \gamma, q')$ from state (q, x) to state (q', x), while updating the counter from z to $z + d_x - d_y$, and gaining an immediate reward of d_y. Two special states called *accrue* and *stop* model the termination of the MDPAO. We restrict the analysis to non-negative register updates so that the rewards in the OC-MDP will be non-negative too.

Formally, given a 2-register MDPAO $\mathcal{M}$, we define the corresponding OC-MDP $\mathcal{A}$ as follows. The set of control states S comprises pairs (q, r), where $q \in Q$ and $r \in \{x, y\}$,



action	$p(x)$	$p(y)$
λ_1	ε	ε
λ_2	ε	ε
β	$\frac{1}{4}$	$\frac{1}{4}$
γ	ε	0
γ'	1	0

Figure 4 A 2-register MDPAO. Edges are labeled with an action name followed by the update to variable x. The probabilities of terminating with x or y for each action are reported in the table.

plus the special states *accrue* and *stop*. The actions are the same as in the MDPAO. For all $q, q' \in Q$, $r \in \{x, y\}$, and $\gamma \in \Gamma$, the following transitions belong to Δ_0 (recall that $\delta(\cdot,\cdot,\cdot) \in \{0,1\}$):

$((q,r), \gamma, p(q,\gamma,q'), \delta(q,\gamma,x) - \delta(q,\gamma,y), \delta(q,\gamma,y), (q',x))$ whenever $\delta(q,\gamma,x) \geq \delta(q,\gamma,y)$

$((q,r), \gamma, p(q,\gamma,q'), \delta(q,\gamma,y) - \delta(q,\gamma,x), \delta(q,\gamma,x), (q',y))$ whenever $\delta(q,\gamma,x) < \delta(q,\gamma,y)$

$(accrue, \gamma, 1, 0, 0, stop)$

Similarly, the following transitions belong to $\Delta_{>0}$:

$((q,x), \gamma, p(q,\gamma,q'), \delta(q,\gamma,x) - \delta(q,\gamma,y), \delta(q,\gamma,y), (q',x))$
$((q,y), \gamma, p(q,\gamma,q'), \delta(q,\gamma,y) - \delta(q,\gamma,x), \delta(q,\gamma,x), (q',y))$
$(accrue, \gamma, 1, -1, 1, accrue)$
$(stop, \gamma, 1, -1, 0, stop)$

Moreover, both in Δ_0 and in $\Delta_{>0}$ we find the following transitions, where $\neg r$ denotes the register different from r:

$((q,r), \gamma, p(q,\gamma,r), 0, 0, accrue)$ $\qquad$ $((q,r), \gamma, p(q,\gamma,\neg r), 0, 0, stop)$

As an example, consider the 2-register MDPAO in Figure 4. To keep the figure readable, we do not draw the edges connecting each state to the exit nodes x and y, except for action γ' from state q_3, because the only effect of γ' is to exit with value x. The corresponding probabilities are reported in the table on the r.h.s. of the figure. No action modifies register y, whereas updates to register x appear next to the action name. As an example, for action β taken from q_2 we have $p(q_2,\beta,x) = 1/4$, $p(q_2,\beta,y) = 1/4$, and $p(q_2,\beta,q_1) = 1/2$, with register updates $\delta(q_2,\beta,x) = 1$ and $\delta(q_2,\beta,y) = 0$.

Figure 5 shows a fragment of the corresponding OC-MDP, starting from state (q_2, x), representing the original state q_2 when $y = 0$. Termination of the original MDP in register x is simulated by moving to *accrue*. In that state, the counter is repeatedly decremented until it reaches 0, with each iteration adding 1 to the reward. When the counter reaches 0, the OC-MDP moves to the final state *stop*. Termination in register y is simulated by moving directly to *stop*, as no further reward needs to be accrued, because we are assuming that $y = 0$ and the counter contains the value of register x.

▶ **Theorem 8.** *Let $\mathcal{M}$ be a 2-register MDPAO with non-negative register updates. The minimum expected cost $f(q,0,0)$ in $\mathcal{M}$ is equal to the minimum expected accumulated reward in $\mathcal{A}$ from $((q,x),0)$ to $(stop,0)$.*

The minimum expected accumulated reward problem is a generalization of the minimum expected termination time problem considered in [8]. Neither problem is known to be

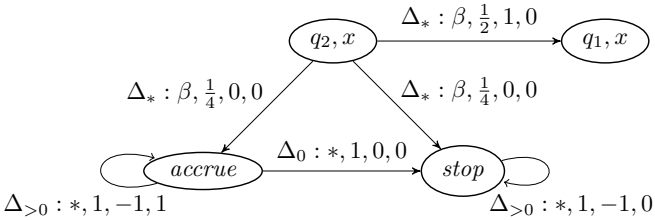


Figure 5 A fragment of the OC-MDP corresponding to the MDPAO in Figure 4. Edges report action, probability, counter update, and reward. The asterisk is used as a wildcard.

decidable for OC-MDPs, whereas both are solvable in PSPACE for probabilistic 1-counter automata (p1CAs) [9, 7], which are equivalent to OC-MDPs with a single action. In Subsection 6.3, we employ the decidability on p1CAs to solve the minimum expected cost problem on a class of 2-register MDPAOs.

6.2 Tie-less MDPAOs

In this subsection, we introduce the class of *tie-less* MDPAOs and show that for this class there exists an optimal strategy that is eventually counter-oblivious.

Note that from equation (2), $f(q, x, y)$ can be computed from $f(q, x - y, 0)$ and thus it suffices to compute $f(q, n, 0)$, with $n \in \mathbb{Z}$. In the following we write $f(q, n)$ as a shortcut for $f(q, n, 0)$. A *simple strategy* is a function $\sigma : Q \to \Gamma$ and we denote by $\mathcal{S}$ the set of all simple strategies. An *ω-strategy* is an infinite sequence of simple strategies, i.e., an element of $\mathcal{S}^\omega$. Notice that given a full strategy σ, an initial state q and an initial valuation ν there exists at least one ω-strategy π that is equivalent to it. In particular, it holds $f_\sigma(q, \nu) = f_\pi(q, \nu)$. Hence, in order to compute $f(q, 0, 0)$ it is sufficient to consider ω-strategies rather than full strategies.

Similarly to Lemma 4, we can show that for all ω-strategies π, $f_\pi(q, n)$ is a linear function of n, i.e.: $f_\pi(q, n) = \mathit{slope}_{q,x}(\pi) \cdot n + f_\pi(q, 0)$, where $\mathit{slope}_{q,x}(\pi)$ is the probability that a path starting at q ends in register x under strategy π. Notice that $\mathit{slope}_{q,x}(\pi)$ generalizes the notation $a_{q,x}$ used in Section 4 to denote the probability of terminating with a specific register.

In fact, if the same simple strategy $\sigma \in \mathcal{S}$ is used at every step (equivalently, the ω-strategy σ^ω is used), the system becomes equivalent to an MDPAO with a single action and therefore $\mathit{slope}_{q,x}(\sigma^\omega)$ can be computed for all q in polynomial time as explained in Section 4. In particular, the simple strategy α with the minimum x-slope can be found by solving the following linear program, on the set of variables $\{a_q \mid q \in Q\}$.

$$\begin{aligned} &\text{maximize:} && \sum_{q \in Q} a_q && (3)\\ &\text{subject to:} && a_q \le p(q, \gamma, x) + \sum_{q' \in Q} p(q, \gamma, q') a_{q'} \quad \forall\, q \in Q, \gamma \in \Gamma \end{aligned}$$

The optimal solution a^* provides the minimal x-slope in each state. We say that the MDPAO is *x-tie-less* if there exists a unique simple strategy α that has the minimum x-slope from all states q. Equivalently, the MDPAO is x-tie-less if at a^* exactly one of the constraints

for each state is tight, i.e., for all q there exists a unique $\alpha_q \in \Gamma$ such that

$$a^*_q = p(q, \alpha_q, x) + \sum_{q' \in Q} p(q, \alpha_q, q') a^*_{q'}. \tag{4}$$

We can similarly define the y-slope of a strategy as the probability of terminating in register y using that strategy. Since runs terminate with probability one, for each strategy the sum of the x-slope and the y-slope is 1. We then say that the MDPAO is *y-tie-less* if there exists a unique simple strategy that has the minimum y-slope (or equivalently the maximum x-slope) from all states. Finally, the MDPAO is *tie-less* if it is both x-tie-less and y-tie-less.

Notice that if an MDPAO is not tie-less then it can be made tie-less by an arbitrarily small perturbation of its transition probabilities. In contrast, a tie-less MDPAO can be perturbed by a sufficiently small amount and still be tie-less. So, being tie-less is a *robust* property of MDPAOs, whereas its opposite is not. In the following Lemmas 9-12 we assume that the MDPAO is x-tie-less and α is the simple strategy with minimum x-slope from all states.

By inspecting the linear program (3), we observe that, for all $q \in Q$, the slope of using α indefinitely cannot be improved (i.e., lowered) by starting with any other action. This is formalized by the following result.

▶ **Lemma 9.** *For all $\sigma \in \mathcal{S}$ and $q \in Q$, we have that $slope_{q,x}(\sigma\alpha^\omega) \geq slope_{q,x}(\alpha^\omega)$.*

The following lemma extends the previous one from simple strategies to arbitrary ω-strategies.

▶ **Lemma 10.** *For all $\pi \in \mathcal{S}^\omega$ and $q \in Q$, we have that $slope_{q,x}(\pi) \geq slope_{q,x}(\alpha^\omega)$.*

Proof. First, we prove the result for an ω-strategy that starts with an arbitrary finite prefix and then switches permanently to α, i.e., a strategy of the form $\pi = \tau\alpha^\omega$, for $\tau \in \mathcal{S}^*$. We proceed by induction on $|\tau|$. If $|\tau| = 0$, the result is trivial. Otherwise, $\tau = \tau_0\tau'$ and the induction hypothesis guarantees that $slope_{q,x}(\tau'\alpha^\omega) \geq slope_{q,x}(\alpha^\omega)$. Then,

$$\begin{aligned}
slope_{q,x}(\tau\alpha^\omega) &= p(q, \tau_0(q), x) + \sum_{q' \in Q} p(q, \tau_0(q), q') slope_{q',x}(\tau'\alpha^\omega) \\
&\geq p(q, \tau_0(q), x) + \sum_{q' \in Q} p(q, \tau_0(q), q') slope_{q',x}(\alpha^\omega) \quad \text{by ind. hyp.} \\
&= slope_{q,x}(\tau_0\alpha^\omega) \\
&\geq slope_{q,x}(\alpha^\omega) \qquad \text{by Lemma 9.}
\end{aligned}$$

Next, consider an arbitrary ω-strategy π. By contradiction, let $q \in Q$ be such that $slope_{q,x}(\alpha^\omega, q) - slope_{q,x}(\pi) = c > 0$. Let $p^0_{\max}$ be the maximum probability of continuation, i.e., $p^0_{\max} = \max_{q \in Q, \gamma \in \Gamma} \big(1 - p(q, \gamma, x) - p(q, \gamma, y)\big)$. Clearly $p^0_{\max} < 1$.

One can easily prove the following **claim:** Let $\pi^1, \pi^2 \in \mathcal{S}^\omega$ be two ω-strategies that coincide on the first k steps. Then, $|slope_{q,x}(\pi^1) - slope_{q,x}(\pi^2)| \leq (p^0_{\max})^k$.

Now, let $n > 0$ be such that $(p^0_{\max})^n < c$ (i.e., $n > \frac{\log c}{\log p^0_{\max}}$). It holds

$$slope_{q,x}(\pi_{\leq n}\alpha^\omega) \leq slope_{q,x}(\pi) + (p^0_{\max})^n < slope_{q,x}(\pi) + c = slope_{q,x}(\alpha^\omega).$$

This contradicts the above argument for finite prefixes and the thesis follows. ◀

The following lemma shows that when the initial value of register x grows, the x-slope of any optimal strategy approaches the minimum possible x-slope, i.e., the x-slope of the strategy α^ω. We denote by M an upper bound to $|f_\pi(q, 0)|$, for all π and q, as provided by Lemma 1.

▶ **Lemma 11.** *For all $q \in Q$, let $(\pi^{(k)})_{k\in\mathbb{N}}$ be a sequence of ω-strategies s.t. for all k, $f_{\pi^{(k)}}(q,k) = f(q,k)$ (i.e., $\pi^{(k)}$ is optimal from q and k). We have that $\lim_{k\to\infty} slope_{q,x}(\pi^{(k)}) = slope_{q,x}(\alpha^\omega)$.*

Proof. By Lemma 10, $slope_{q,x}(\pi^{(k)}) \geq slope_{q,x}(\alpha^\omega)$. We prove that for all $\epsilon > 0$ there exists $k > 0$ such that for all $m \geq k$ it holds $slope_{q,x}(\pi^{(m)}) < slope_{q,x}(\alpha^\omega) + \epsilon$. Let $k > \frac{2M}{\epsilon}$ and $m \geq k$. Assume by contradiction that $slope_{q,x}(\pi^{(m)}) \geq slope_{q,x}(\alpha^\omega) + \epsilon$. Then, we have:

$$\begin{aligned} f_{\pi^{(m)}}(q,m) &= slope_{q,x}(\pi^{(m)})m + f_{\pi^{(m)}}(q,0) \\ &\geq slope_{q,x}(\pi^{(m)})m - M \\ &\geq \big(slope_{q,x}(\alpha^\omega) + \epsilon\big)m - M \\ &> slope_{q,x}(\alpha^\omega)m + \epsilon\frac{2M}{\epsilon} - M \\ &= slope_{q,x}(\alpha^\omega)m + M \\ &\geq f_{\alpha^\omega}(q,m). \end{aligned}$$

This contradicts the fact that $\pi^{(m)}$ is optimal from q and m, and the thesis follows. ◀

For a state q, let $\mathcal{S}_q$ be the set of simple strategies σ such that $\sigma(q) \neq \alpha(q)$, and let

$$\begin{aligned} c_q &= \min_{\sigma\in\mathcal{S}_q} slope_{q,x}(\sigma\alpha^\omega) - slope_{q,x}(\alpha^\omega) \\ &= \min_{\gamma\in\Gamma\setminus\{\alpha(q)\}} \Big(p(q,\gamma,x) + \sum_{q'\in Q} p(q,\gamma,q')slope_{q',x}(\alpha^\omega)\Big) - slope_{q,x}(\alpha^\omega). \end{aligned}$$

If $\mathcal{M}$ is x-tie-less then $c_q > 0$ and the following lemma states that for large enough m the optimal strategy $\pi^{(m)}$ starts with the action $\alpha(q)$.

▶ **Lemma 12.** *For all $q \in Q$ and $m > \frac{2M}{c_q}$, let $\pi^{(m)}$ be an ω-strategy that is optimal from q and m. We have that $\pi_0^{(m)}(q) = \alpha(q)$.*

Proof. According to the proof of Lemma 11, for all $m > \frac{2M}{c_q}$ it holds $slope_{q,x}(\pi^{(m)}) < slope_{q,x}(\alpha^\omega) + c_q$. Assume by contradiction that there exists m such that $\pi_0^{(m)}(q) = \gamma \neq \alpha(q)$. Then, we have

$$\begin{aligned} slope_{q,x}(\pi^{(m)}) &= p(q,\gamma,x) + \sum_{q'\in Q} p(q,\gamma,q')slope_{q',x}(\pi_{\geq 1}^{(m)}) \\ &\geq p(q,\gamma,x) + \sum_{q'\in Q} p(q,\gamma,q')slope_{q',x}(\alpha^\omega) \quad \text{by Lemma 10} \\ &\geq slope_{q,x}(\alpha^\omega) + c_q \quad \text{by def. of } c_q. \end{aligned}$$

This is a contradiction and the thesis follows. ◀

A series of symmetrical arguments shows that for $m \ll 0$, the optimal strategies from a configuration $(q,m,0)$ start with the simple strategy that has the *maximum* x-slope, or equivalently the minimum y-slope. By combining Lemma 12 and its symmetrical counterpart for negative m, we obtain the following, where a path-oblivious strategy σ is said to also be *counter-oblivious beyond* m if for all $m_1, m_2 \geq m$ and all $q \in Q$, it holds $\sigma(q,m_1,0) = \sigma(q,m_2,0)$ and $\sigma(q,-m_1,0) = \sigma(q,-m_2,0)$.

▶ **Theorem 13.** *For all tie-less 2-register MDPAOs, we can compute in polynomial time a number m such that there exists a strategy that is counter-oblivious beyond m and optimal.*

In the following subsection, Theorem 13 is used to prove computability of the minimum expected cost.

To conclude this subsection, we show that the property of Lemma 12 does not hold for general 2-register MDPAOs. The already mentioned MDPAO in Figure 4 is such that the optimal strategy is *not* eventually counter-oblivious. This example is inspired by the proof that approximating the minimum expected termination time of an OC-MDP is computationally hard [8]. Notice that said MDPAO is not x-tie-less, because both simple strategies $\sigma_1 = \{q_0 \to \lambda_1, q_1 \to \beta, q_2 \to \beta, q_3 \to \gamma'\}$ and $\sigma_2 = \{q_0 \to \lambda_2, q_1 \to \beta, q_2 \to \beta, q_3 \to \gamma'\}$ achieve the minimal x-slope from each state.

When starting from q_1 or q_2, with register values $x \ll 0$ and $y = 0$, the optimal strategy consists in staying in the q_1q_2 loop for some time and then eventually exiting the loop by choosing action γ in q_1. This is because the x-slope of the strategy that stays in the loop is $1/2$, smaller than the x-slope of the strategy that exits the loop, which is 1. Later, when the value of x gets close to 0 and then positive, it becomes convenient to exit the loop, increase x by d and pay the current value of x. Moreover, depending on parameters d and ε, there is a specific value k for x at which it is maximally convenient to exit the loop. If the system is in q_1 when $x = k$, then the optimal strategy picks γ and exits the loop. If instead the system is in q_2, the strategy cannot immediately exit from the loop, so it must either exit the loop at $x = k - 1$ or at $x = k + 1$, whichever gives the least cost.

When starting in q_0, the optimal move is the one that ensures that the system will be in q_1 when x strikes the critical value k. Specifically, one can check that for sufficiently small ε and for $d \in (5, 5.5)$ [2] we obtain $k = 8$, so that the optimal move from $(q_0, -n, 0)$ is λ_1 when n is even and λ_2 when n is odd.

6.3 From tie-less MDPAOs to probabilistic 1-counter automata

Given a tie-less 2-register MDPAO $\mathcal{M}$ and a state q, we reduce its minimum expected cost problem from $(q, 0, 0)$ to a finite number of expected accumulated reward problems for probabilistic 1-counter automata (p1CA), each of which is decidable via an encoding in the existential Theory of Reals [7]. A p1CA is essentially an OC-MDP with a single action, also equivalent to a probabilistic pushdown automaton with a single stack symbol.

By Theorem 8, let $\mathcal{A} = (S, \Gamma, \Delta_0, \Delta_{>0})$ be the OC-MDP equivalent to $\mathcal{M}$. Let n be the threshold provided by Theorem 13 for $\mathcal{M}$. When looking for an optimal strategy for $\mathcal{M}$, we can limit our search to strategies that are path-oblivious and counter-oblivious beyond n. The set of all such strategies has cardinality $|\Gamma|^{|Q|\cdot n}$, which is doubly exponential in the size of the original MDPAO.

Given such a strategy σ, we build a single-action OC-MDP $\mathcal{A}_\sigma = (S', \{\gamma\}, \Delta'_0, \Delta'_{>0})$, whose expected accumulated reward from a distinguished state is equal to the expected cost of σ from $(q, 0, 0)$. The automaton $\mathcal{A}_\sigma$ is obtained from $\mathcal{A}$ as follows:

- By embedding the counter values $\{0, \dots, n\}$ into the state, i.e., $S' = S \times \{0, \dots, n\}$, with the intended meaning that each enlarged state $\langle s, k\rangle$, with $k < n$, is only visited with counter value 0 and corresponds to state s with counter value k in $\mathcal{A}$, whereas each enlarged state $\langle s, n\rangle$ with counter value l corresponds to state s and counter value $n + l$ in $\mathcal{A}$.

[2] A rational register update d can be easily simulated by probabilistically inducing the appropriate convex combination of the integer updates $\lfloor d \rfloor$ and $\lceil d \rceil$.

- By modifying the transitions according to the above encoding, while retaining only the actions chosen by the strategy σ. For instance, consider the enlarged state $\langle(q,x),k\rangle \in S'$, with $0 < k < n$, and let $\alpha = \sigma(q,k,0)$. Assume that the following transition occurs in $\mathcal{A}$: $((q,x),\alpha,p,d,r,(q',x)) \in \Delta_{>0}$. Then, the following occurs in $\mathcal{A}_\sigma$: $\big(\langle(q,x),k\rangle,\gamma,p,0,$ $r,\langle(q',x),k+d\rangle\big) \in \Delta'_0$.
  Notice that under our assumptions, $d \in \{-1,0,1\}$ and so $k+d \in \{0,\ldots,n\}$. On the other hand, $\Delta'_{>0}$ contains no transitions starting from $\langle(q,x),k\rangle$, because that state is only intended to be visited with counter value zero.

It is easy to prove by construction that the expected accumulated reward from $\langle(q,x),0\rangle$ in $\mathcal{A}_\sigma$ is equal to the expected cost of σ from $(q,0,0)$ in $\mathcal{M}$. Hence, we obtain the following.

▶ **Theorem 14.** *The minimum expected cost problem for tie-less 2-register MDPAOs with non-negative register updates is decidable in 2EXPTIME.*

7 Approximation algorithm for MDPs with alternative objectives

While computing the minimum expected cost even for simple models proves to be difficult, it is possible to compute the minimum expected cost in a general MDPAO $\mathcal{M}$ up to an additive error ϵ given $q \in Q$ and $\nu \in \mathcal{V}$. The idea is to compute the minimum expected cost of paths of length at most k and for large values of k, we can show that it is close to the actual minimum expected cost. Let $f^k(q,\nu)$ be this cost, i.e. $f^k(q,\nu) = \min_\sigma \sum_{\rho\in\Pi^k_\sigma(q,\nu)} Pr(\rho)f(\rho)$ where $\Pi^k_\sigma(q,\nu)$ is the set of paths of length at most k that start in (q,ν) and are consistent with σ.

Now, we show a result in Lemma 15 that bounds the difference between the actual cost $f(q,\nu)$ and $f^k(q,\nu)$ for any positive k. Let δ_M be the maximum change to a register in a step in the process and p_M be the maximum probability of continuation as defined in Lemma 1.

▶ **Lemma 15.** *Given a state $q \in Q$ and $\nu \in \mathcal{V}$ in an MDPAO $\mathcal{M}$,*

$$-p_M^k B(\max_{x\in X}|\nu(x)| + k\delta_M) \;\le\; f(q,\nu) - f^k(q,\nu) \;\le\; p_M^k B(\max_{x\in X}|\nu(x)| + k\delta_M) \;\triangleq z,$$

where B is a function of δ_M and p_M is described in Lemma 1.

Proof. To prove the lower bound, we split the cost $f(q,\nu)$ into two components f_1^k, which is the cost paid in the first k steps, and f_{k+1}^∞, the cost paid in the remaining steps. Note that $f_1^k \ge f^k(q,\nu)$, since $f^k(q,\nu)$ is the minimum expected cost of k steps. Further, after k steps, the probability that the process does not terminate is at most p_M^k and the register values are at least $-(\max_{x\in X}|\nu(x)| + k\delta_M)$. Thus, using Lemma 1, we can see that $f_{k+1}^\infty > -z$ which gives the required lower bound. To show the upper bound, we consider a strategy σ, where the first k actions minimize the expected cost of paths of length k, and the subsequent actions are arbitrary. By a similar analysis as above, $f_\sigma(q,\nu) \le f^k(q,\nu) + z$. Also, $f(q,\nu) \le f_\sigma(q,\nu)$ and hence, we have the required upper bound. ◀

Now, if we choose k such that $z = \epsilon$, the difference $|f(q,\nu) - f^k(q,\nu)|$ is bounded by ϵ and thus, k is $O(|\frac{\log\epsilon}{\log p_M}|)$. To compute $f^k(q,\nu)$, we transform $\mathcal{M}$ into an MDP $\mathcal{M}'$ with a node for each state and register valuation possible on paths of length at most k, starting from (q,ν). Note that there are at most $2k\delta_M + 1$ values possible for each register. Therefore, total number of states in $\mathcal{M}'$ is at most $O(|Q|(k\delta_M)^{|X|})$. We use dynamic programming to compute $f^k(q,\nu)$ by computing the minimum expected cost for $i+1$ steps using the cost for i steps. We can see that time to compute $f^k(q,\nu)$ is $(|\Gamma||Q|(|\frac{\log\epsilon}{\log p_M}|\delta_M)^{O(|X|)})$.

▶ **Theorem 16.** *In an MDPAO $\mathcal{M}$, the minimum expected cost $f(q,\nu)$ can be computed up to an additive error ϵ in $|\Gamma||Q|(|\frac{\log \epsilon}{\log p_M}|\delta_M)^{O(|X|)}$ time.*

8 Conclusions

We have introduced the model of Markov decision processes with alternative objectives to analyze situations where there are a number of alternative cost/reward objectives of which only a single one is actualized upon termination. We believe that the formalization and our results will find practical applications to planning scenarios with uncertain future rewards.

From a theoretical viewpoint, compared with the existing literature on MDPs, the optimization problem we have considered has an unusual structure worthy of further research: the underlying process is finite-state but the optimal choice depends on the infinite set of cumulative costs. Finding an exact solution for the general case remains an open problem, and as a next step, we would like to investigate whether it is always the case that two-register MDPAOs admit optimal strategies that are eventually periodic (see Section 6).

References

1. R. Alur, L. D'Antoni, J. Deshmukh, M. Raghothaman, and Y. Yuan. Regular functions and cost register automata. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 13–22, 2013.
2. R. Alur and M. Raghothaman. Decision problems for additive regular functions. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP, Part II*, pages 37–48, 2013.
3. R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.
4. D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. Oper. Res.*, 16(3):580–595, August 1991.
5. T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. In *International Colloquium on Automata, Languages, and Programming*, pages 332–343, 2011.
6. T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-counter Markov decision processes. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 863–874, 2010.
7. T. Brázdil, J. Esparza, S. Kiefer, and A. Kučera. Analyzing probabilistic pushdown automata. *Form. Methods Syst. Des.*, 43(2):124–163, October 2013.
8. T. Brázdil, A. Kučera, P. Novotný, and D. Wojtczak. Minimizing expected termination time in one-counter Markov decision processes. In *Automata, Languages, and Programming – 38th ICALP, Part II*, pages 141–152, 2012.
9. J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: expectations and variances. In *Proceedings of the 2005 20th Annual IEEE Symposium on Logic in Computer Science*, pages 117–126, 2005.
10. E. A. Feinberg and A. Shwartz. *Handbook of Markov decision processes: methods and applications*, volume 40. Springer Science & Business Media, 2012.
11. M. Kwiatkowska. Quantitative verification: Models, techniques and tools. In *Proc. ACM SIGSOFT Symp. on Foundations of Software Engineering*, pages 449–458, 2007.
12. D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Int. Res.*, 48(1):67–113, October 2013.

Minimizing Regret in Discounted-Sum Games*

Paul Hunter[1], Guillermo A. Pérez[†2], and Jean-François Raskin[3]

1 **Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium**
phunter@ulb.ac.be
2 **Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium**
gperezme@ulb.ac.be
3 **Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium**
jraskin@ulb.ac.be

Abstract

In this paper, we study the problem of minimizing regret in discounted-sum games played on weighted game graphs. We give algorithms for the general problem of computing the minimal regret of the controller (Eve) as well as several variants depending on which strategies the environment (Adam) is permitted to use. We also consider the problem of synthesizing regret-free strategies for Eve in each of these scenarios.

1998 ACM Subject Classification F.1.1 Automata, D.2.4 Formal methods

Keywords and phrases Quantitative games, Regret, Verification, Synthesis, Game theory

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.30

1 Introduction

Two-player games played by Eve and Adam on weighted graphs is a well accepted mathematical formalism for modelling quantitative aspects of a controller (Eve) interacting with its environment (Adam). The outcome of the interaction between the two players is an infinite path in the weighted graph and a value is associated to this infinite path using a *measure* such as *e.g.* the mean-payoff of the weights of edges traversed by the infinite path, or the discounted sum of those weights. In the classical model, the game is considered to be zero sum: the two players have antagonistic goals–one of the player want to maximize the value associated to the outcome while the other want to minimize this value. The main solution concept is then the notion of winning strategy and the main decision problem asks, given a threshold c, whether Eve has a strategy to ensure that, no matter how Adam plays, that the outcome has a value larger than or equal to c.

When the environment is not fully antagonistic, it is reasonable to study other solution concepts. One interesting concept to explore is the concept of *regret minimization* [3] which is as follows. When a strategy of Adam is fixed, we can identify the set of Eve's strategies that allow her to secure the *best possible outcome* against this strategy. This constitutes Eve's *best response*. Then we define the regret of a strategy σ of Eve as the difference between Eve's best response; and the payoff she secures thanks to her strategy σ. So, when trying to minimize the regret associated to a strategy, we use best responses as a *yardstick*. Let us now illustrate this with an example.

* This work was supported by the ERC inVEST (279499) project.
† Author supported by F.R.S.-FNRS fellowship.

© Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin;
licensed under Creative Commons License CC-BY

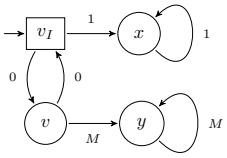


Figure 1 A game in which waiting is required to minimize regret.

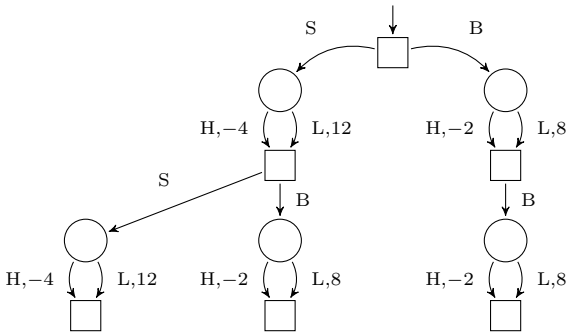


Figure 2 A game that models different investment strategies.

Table 1 The possible rate configurations for the rate of interests are given as the first four columns, then follows the worst-case performance and the regret associated to each strategy of Eve that are given in rows. Entries in bold are the values that are maximizing the worst case (strategy BB) and minimizing the regret (strategy SB).

	HH	HL	LH	LL	Worst-case	Regret
SS	−7.7616	7.6048	7.9784	23.2848	−7.7616	3.8808
SB	−5.8408	3.7632	9.8392	19.4432	−5.8408	**3.8416**
BB	−3.8808	5.7232	5.9192	15.5232	**−3.8808**	7.7616

▶ **Example 1** (Investment advice)**.** Consider the discounted sum game of Fig. 2. It models the rentability of different investment plans with a time horizon of two periods. In the first period, it can be decided to invest in treasure bonds (B) or to invest in the stock market (S). In the former case, treasure bonds (B) are chosen for two periods. In the latter case, after one period, there is again a choice for either treasure bonds (B) or stock market (S). The returns of the different investments depend on the fluctuation of the rate of interests. When the rate of interests is low (L) then the return for the stock market investments is equal to 12 and for the treasure bonds it is equal to 8. When the interest rate is high (H) then the returns for the stock market investments is equal to −4 and for the treasure bonds it is equal to −2. To model time and take into account the inflation rate, say equal to 2 percent, we consider a discount factor $\lambda = 0.98$ for the returns. In this example, we make the hypothesis that the fluctuation of the rate of interests is *not* a function of the behavior of the investor. It means that this fluctuation rate is either one of the following four possibilities: HH, HL, LH, LL. This corresponds to Adam playing a *word strategy* in our terminology. The discounted sum of returns obtained under the 12 different scenarios are given in Table 1.

Now, assume that you are a broker and you need to advise one of your customers regarding his next investment. There are several ways to advise your customer. First, if your customer is strongly *risk averse*, then you should be able to convince him that he has to go for the treasure bonds (B). Indeed, this is the choice that *maximizes the worst case*: if the interest rates stay high for two periods (HH) then the loss will be −3.8808 while it will be higher for any other choices. Second, and maybe more interestingly, if your customer tolerates some risks, then you may want to keep him happy so that he will continue to ask for your advice in the future! Then you should propose the following strategy: first invest in the stock market (S) then in treasure bonds (B) as this strategy *minimizes regret*. Indeed, at the end of the two investment periods, the actual interest rates will be known and so your customer will

evaluate your advices *ex-post*. So, after the two periods, the value of the choices made *ex ante* can be compared to the best strategy that could have been chosen knowing the evolution of the interest rates. The regret of SB is at most equal to 3.8416 in all cases and it is minimal: the regret of BB can be as high as 7.7616 if LL is observed, and the regret of SS can be as high as 3.8808.

Finally, let us remark that if the investments are done in financial markets that are subject to different interest rates, then instead of considering the minimization of regret against word strategies, then we could consider the regret against all strategies. We also study this case in this paper. ◀

Previous works. In [8], we studied regret minimization in the context of reactive synthesis for shortest path objectives. Recently in [13], we studied the notion of regret minimization when we assume different sets of strategies from which Adam chooses. We have considered three cases: when the Adam is allowed to play *any* strategy, when he is restricted to play a *memoryless* strategy, and when he plays *word* strategies. We refer the interested reader to [13] for motivations behind each of these definitions. In that paper, we studied the regret minimization problem for the following classical quantitative measures: inf, sup, $\liminf$, $\limsup$ and the *mean-payoff* measure. In this paper, we complete this picture by studying the regret minimization problem for the *discounted-sum* measure. Discounted-sum is a central measure in quantitative games but we did not consider it in [13] because it requires specific techniques which are more involved than the ones used for the other quantitative measures. For example, while for mean-payoff objectives, strategies that minimize regret are memoryless when Adam can play any strategy, we show in this paper that pseudo-polynomial memory is necessary (and sufficient) to minimize regret in discounted-sum games. The need for memory is illustrated by the following example.

▶ **Example 2.** Consider the example in Figure 1 where $M \gg 1$. Eve can play the following strategies in this game: let $i \in \mathbb{N} \cup \{\infty\}$, and note σ^i the strategy that first plays i rounds the edge (v_I, v) and then switches to (v_I, x). The regret values associated to those strategies are as follows.

The regret of σ^∞ is $\frac{1}{1-\lambda}$ and it is witnessed when Adam never plays the edge (v, y). Indeed, the discounted sum of the outcome in that case is 0, while if Eve had chosen to play (v_I, x) at the first step instead, then she would have gained $\frac{1}{1-\lambda}$. The regret of σ^i is equal to the maximum between $\frac{1}{1-\lambda} - \lambda^{2i}\frac{1}{1-\lambda}$ and $\lambda^{2i+1}\frac{M}{1-\lambda} - \lambda^{2i}\frac{1}{1-\lambda}$. The maximum is either witnessed when Adam never plays (v, y) or plays (v, y) if the edge (v_I, x) has been chosen $i+1$ times (one more time compared to σ^i).

So the strategy that minimizes regret is the strategy σ^N for $N > \frac{-\log M}{2\log\lambda} - \frac{1}{2}$ (so that $\lambda^{2N+1}M < 1$), *i.e.* the strategy needs to count up to N. ◀

Contributions. We describe algorithms to decide the regret threshold problem for games in three cases: when there is no restriction on the strategies that Adam can play, when Adam can only play memoryless strategies, and when Adam can only play word strategies. For this last case, our problem is closely related to open problems in the field of discounted-sum automata, and we also consider variants given as ε-gap promise problems.

We also study the complexity of the special case when the threshold is 0, *i.e.* when we ask for the existence of regret free strategies. We show that problem is sometimes easier to solve. Our results on the complexity of both the regret threshold and the regret-free problems are summarized in Table 2. All our results are for fixed discount factor λ.

Table 2 Complexity of deciding the regret threshold and regret-free problems for fixed λ.

	Any strategy	Memoryless strategies	Word strategies
regret threshold	NP (Thm. 8)	PSPACE (Thm. 20), coNP-h (Thm. 26)	PSPACE-c (ε-gap) (Thm. 29, Thm. 30)
regret-free	PTIME (Thm. 9)	PSPACE (Thm. 22), coNP-h (Thm. 26)	NP-c (Thm. 27)

Other related works. A Boolean version of our *regret-free strategies* has been described in [7]. In that paper, they are called *remorse-free strategies*. These correspond to strategies that ensure a regret value of 0 in games with ω-regular objectives. They do not establish lower bounds on the complexity of realizability or synthesis of remorse-free strategies and they only consider word strategies for Adam.

In [13], we established that regret minimization when Adam plays word strategies only is a generalization of the notion of *good-for-games automata* [10] and *determinization by pruning* (of a refinement) [1].

The notion of regret is closely related to the notion of competitive ratios used for the analysis of online algorithms [14]: the performance of an online algorithm facing uncertainty (*e.g.* about the future incoming requests or data) is compared to the performance of an offline algorithm (where uncertainty is resolved). According to this quality measure, an online algorithm is better if its performance is closer to the performance of an optimal offline solution.

Structure of the paper. In Sect. 2, we introduce the necessary definitions and notations. In Sect. 3, we study the minimization of regret when the second player plays any strategy. Finally, in Sect. 4, we study the minimization of regret when the second player plays a memoryless strategy and in Sect. 5 when he plays a word strategy.

For reasons of space, some claims presented in the sequel are only supported by a short proof sketch. We refer the interested reader to the technical report [12] for the full proofs.

2 Preliminaries

A *weighted arena* is a tuple $G = (V, V_\exists, E, w, v_I)$ where (V, E, w) is an edge-weighted graph (with rational weights), $V_\exists \subseteq V$, and $v_I \in V$ is the initial vertex. For a given $u \in V$ we denote by $\mathbf{succ}(u)$ the set of *successors of* u *in* G, that is the set $\{v \in V : (u, v) \in E\}$. We assume w.l.o.g. that no vertex is a sink, *i.e.* $\forall v \in V : |\mathbf{succ}(v)| > 0$, and that every Eve vertex has more than one successor, *i.e.* $\forall v \in V_\exists : |\mathbf{succ}(v)| > 1$. In the sequel, we depict vertices in $V_\exists$ with squares and vertices in $V \setminus V_\exists$ with circles. We denote the maximum absolute value of a weight in a weighted arena by $W := \max_{e \in E} |w(e)|$.

A *play* in a weighted arena is an infinite sequence of vertices $\pi = v_0 v_1 \dots$ where $(v_i, v_{i+1}) \in E$ for all $i \geq 0$. Given a play $\pi = v_0 v_1 \dots$ and integers $k \leq l$ we define $\pi[k..l] := v_k \dots v_l$, $\pi[..k] := \pi[0..k]$, and $\pi[l..] := v_l v_{l+1} \dots$. We refer to the latter two as *play prefixes* and *play suffixes*, respectively. To improve readability, we try to adhere to the following convention: use π to denote plays and ρ for play prefixes. The *length of a play* π, denoted $|\pi|$, is ∞, and the *length of a play prefix* $\rho = v_0 \dots v_n$, *i.e.* $|\rho|$, is $n + 1$.

A *strategy for* Eve (Adam) is a function σ that maps play prefixes ending with a vertex v from $V_\exists$ ($V \setminus V_\exists$) to a successor of v. A strategy has memory m if it can be realized as the output of a finite state machine with m states (see *e.g.* [11] for a formal definition). A

memoryless (or positional) strategy is a strategy with memory 1, that is, a function that only depends on the last element of the given partial play. A play $\pi = v_0 v_1 \dots$ is *consistent with a strategy* σ for Eve (Adam) if whenever $v_i \in V_\exists$ ($v_i \in V \setminus V_\exists$), then $\sigma(\pi[..i]) = v_{i+1}$. We denote by $\mathfrak{S}_\exists(G)$ ($\mathfrak{S}_\forall(G)$) the set of all strategies for Eve (Adam) and by $\Sigma^m_\exists(G)$ ($\Sigma^m_\forall(G)$) the set of all strategies for Eve (Adam) in G that require memory of size at most m, in particular $\Sigma^1_\exists(G)$ ($\Sigma^1_\forall(G)$) is the set of all memoryless strategies for Eve (Adam) in G. We omit G if the context is clear.

Given strategies σ, τ, for Eve and Adam respectively, and $v \in V$, we denote by $\pi^v_{\sigma\tau}$ the unique play starting from v that is consistent with σ and τ. If v is omitted, it is assumed to be v_I.

A *weighted automaton* is a tuple $\Gamma = (Q, q_I, A, \Delta, w)$ where A is a finite alphabet, Q is a finite set of states, q_I is the initial state, $\Delta \subseteq Q \times A \times Q$ is the transition relation, $w : \Delta \to \mathbb{Q}$ assigns weights to transitions. A *run* of Γ on a word $a_0 a_1 \dots \in A^\omega$ is a sequence $\psi = q_0 a_0 q_1 a_1 \dots \in (Q \times A)^\omega$ such that $(q_i, a_i, q_{i+1}) \in \Delta$, for all $i \geq 0$, and has *value* $\mathbf{Val}(\psi)$ determined by the sequence of weights of the transitions of the run and the payoff function $\mathbf{Val}$. The value Γ assigns to a word x, *i.e.* $\Gamma(x)$, is the supremum of the values of all runs on the word. We say the automaton is deterministic if Δ is functional (*i.e.* for all $q \in Q$ and all $\sigma \in \Sigma$ we have that $|\{q' \in Q : (q, \sigma, q') \in \Delta\}| = 1$.)

Safety games. A *safety game* is played on a weighted arena by Eve and Adam. The goal of Eve is to perpetually avoid traversing edges from a set of *bad edges*, while Adam attempts to force the play through any unsafe edge. More formally, a safety game is a tuple (G, B) where $G = (V, V_\exists, E, v_I)$ is a weighted arena and $B \subseteq E$ is the set of bad edges. A play $\pi = v_0 v_1 \dots$ is *winning for Eve* if $(v_i, v_{i+1}) \notin B$, for all $i \geq 0$, and it is *winning for Adam* otherwise. A strategy for Eve (Adam) is winning for her (him) in the safety game if all plays consistent with it are winning for her (him). A player wins the safety game if (s)he has a winning strategy.

▶ **Lemma 3** (from [2]). *Safety games are* positionally determined*: either Eve has a positional winning strategy or Adam has a positional winning strategy. Determining the winner in a safety game is decidable in linear time.*

Discounted-sum. A play in a weighted arena, or a run in a weighted automaton, induces an infinite sequence of weights. We define below the *discounted-sum* payoff function which maps finite and infinite sequences of rational weights to real numbers. In the sequel we refer to a weighted arena together with a payoff function as a *game*. Formally, given a sequence of weights $\chi = x_0 x_1 \dots$ of length $n \in \mathbb{N} \cup \{\infty\}$, the *discounted-sum* is defined for a rational discount factor $\lambda \in (0, 1)$ as follows: $\mathsf{DS}_\lambda(\chi) := \sum_{i=0}^{n} \lambda^i x_i$. For convenience, we apply payoff functions directly to plays, runs, and prefixes. For instance, given a play or play prefix $\pi = v_0 v_1 \dots$ we write $\mathsf{DS}_\lambda(\pi)$ instead $\mathsf{DS}_\lambda(w(v_0, v_1) w(v_1, v_2) \dots)$.

Consider a fixed weighted arena G, and a discounted-sum payoff function $\mathbf{Val} = \mathsf{DS}_\lambda$ for some $\lambda \in (0, 1)$. Given strategies σ, τ, for Eve and Adam respectively, and $v \in V$, we denote the value of $\pi^v_{\sigma\tau}$ by $\mathbf{Val}^v_G(\sigma, \tau) := \mathbf{Val}(\pi^v_{\sigma\tau})$. We omit G if it is clear from the context. If v is omitted, it is assumed to be v_I.

Antagonistic & co-operative values. Two values associated with a weighted arena that we will use throughout are the *antagonistic and co-operative values*, defined for plays from a vertex $v \in V$ as:

$$\mathbf{aVal}^v(G) := \sup_{\sigma \in \mathfrak{S}_\exists} \inf_{\tau \in \mathfrak{S}_\forall} \mathbf{Val}^v(\sigma, \tau) \qquad \mathbf{cVal}^v(G) := \sup_{\sigma \in \mathfrak{S}_\exists} \sup_{\tau \in \mathfrak{S}_\forall} \mathbf{Val}^v(\sigma, \tau).$$

Again, if G is clear from the context it will be omitted, and if v is omitted it is assumed to be v_I.

We note that, as memoryless strategies are sufficient in discounted-sum games [15], determining if $\mathbf{aVal}$ is larger (or smaller) than a given threshold is decidable in $\mathsf{NP} \cap \mathsf{coNP}$. Furthermore, the values $\mathbf{cVal}$ and $\mathbf{aVal}$ are representable using a polynomial number of bits. In [15] it is also shown that $\mathbf{aVal}$ can be computed in time polynomial (in $\frac{1}{1-\lambda}$, $|V|$, and $\log_2 W$). If λ is given in binary as part of the input, this becomes exponential (in the size of the input). Regardless of whether λ is part of the input, $\mathbf{cVal}$ is computable in polynomial time.

A useful observation used by Zwick and Paterson in [15], and which is implicitly used throughout this work, is the following.

▶ Remark. For all $u \in V$, $\mathbf{cVal}^u(G) = \max\{w(u,v) + \lambda\mathbf{cVal}^v(G) : (u,v) \in E\}$. For all $u \in V_\exists$, $\mathbf{aVal}^u(G) = \max\{w(u,v) + \lambda\mathbf{aVal}^v(G) : (u,v) \in E\}$. For all $u \in V \setminus V_\exists$, $\mathbf{aVal}^u(G) = \min\{w(u,v) + \lambda\mathbf{aVal}^v(G) : (u,v) \in E\}$.

We say a strategy σ for Eve is *worst-case optimal (maximizing)* from $v \in V$ if it holds that $\inf_{\tau \in \mathfrak{S}_\forall} \mathbf{Val}^v(\sigma,\tau) = \mathbf{aVal}^v(G)$. Similarly, a strategy τ for Adam is *worst-case optimal (minimizing)* from $v \in V$ if it holds that $\sup_{\sigma \in \mathfrak{S}_\exists} \mathbf{Val}^v(\sigma,\tau) = \mathbf{aVal}^v(G)$. Also, a pair of strategies σ, τ for Eve and Adam, respectively, is said to be *co-operative optimal* from $v \in V$ if $\mathbf{Val}^v(\sigma,\tau) = \mathbf{cVal}^v(G)$.

▶ **Lemma 4** (from [15]). *The following hold:*

- *there exists $\sigma \in \Sigma^1_\exists$ which is worst-case optimal maximizing from all $v \in V$,*
- *there exists $\tau \in \Sigma^1_\forall$ which is worst-case optimal minimizing from all $v \in V$,*
- *there are $\sigma \in \Sigma^1_\exists$ and $\tau \in \Sigma^1_\forall$ which are co-operative optimal from all $v \in V$.*

We now recall the definition of a *strongly co-operative optimal* strategy σ for Eve. Intuitively, such a strategy attempts to maximize the co-operative value. Formally, for any play prefix $\rho = v_0 \dots v_n$ consistent with σ, and such that $v_n \in V_\exists$ if $\sigma(\rho) = v'$, then $v' \in \mathbf{cOpt}(v_n)$; where $\mathbf{cOpt}(u) := \{v \in V : (u,v) \in E \text{ and } \mathbf{cVal}^u(G) = w(u,v) + \lambda\mathbf{cVal}^v(G)\}$. Finally, we define a new type of strategy for Eve: *co-operative worst-case optimal* strategies. A strategy is of this type if it attempts to maximize the co-operative value while achieving at least the antagonistic value. More formally, for any play prefix $\rho = v_0 \dots v_n$ consistent with σ, and such that $v_n \in V_\exists$, if $\sigma(\rho) = v'$ then $v' \in \mathbf{wOpt}(v_n)$ and

$$w(v_n, v') + \lambda\mathbf{cVal}^{v'}(G) = \max\{w(v_n, v'') + \lambda\mathbf{cVal}^{v''}(G) : v'' \in \mathbf{wOpt}(v_n)\},$$

where $\mathbf{wOpt}(u) := \{v \in V : (u,v) \in E \text{ and } \mathbf{aVal}^u(G) = w(u,v) + \lambda\mathbf{aVal}^v(G)\}$.

It is not hard to verify that strategies of the above types always exist for Eve.

▶ **Lemma 5.** *There exist strongly co-operative optimal strategies and co-operative worst-case optimal strategies for Eve.*

Regret. Let $\Sigma_\exists \subseteq \mathfrak{S}_\exists$ and $\Sigma_\forall \subseteq \mathfrak{S}_\forall$ be sets of strategies for Eve and Adam respectively. Given $\sigma \in \Sigma_\exists$ we define the *regret of σ in G w.r.t. $\Sigma_\exists$ and $\Sigma_\forall$* as:

$$\mathbf{reg}^\sigma_{\Sigma_\exists,\Sigma_\forall}(G) := \sup_{\tau \in \Sigma_\forall}(\sup_{\sigma' \in \Sigma_\exists} \mathbf{Val}(\sigma',\tau) - \mathbf{Val}(\sigma,\tau)).$$

A strategy σ for Eve is then said to be *regret-free* w.r.t. $\Sigma_\exists$ and $\Sigma_\forall$ if $\mathbf{reg}^\sigma_{\Sigma_\exists,\Sigma_\forall}(G) = 0$. We define the *regret of G w.r.t. $\Sigma_\exists$ and $\Sigma_\forall$* as:

$$\mathbf{Reg}_{\Sigma_\exists,\Sigma_\forall}(G) := \inf_{\sigma \in \Sigma_\exists} \mathbf{reg}^\sigma_{\Sigma_\exists,\Sigma_\forall}(G).$$

When $\Sigma_\exists$ or $\Sigma_\forall$ are omitted from $\mathbf{reg}(\cdot)$ and $\mathbf{Reg}(\cdot)$ they are assumed to be the set of all strategies for Eve and Adam.

In the unfolded definition of the regret of a game, *i.e.*

$$\mathbf{Reg}_{\Sigma_\exists,\Sigma_\forall}(G) := \inf_{\sigma\in\Sigma_\exists} \sup_{\tau\in\Sigma_\forall}(\sup_{\sigma'\in\Sigma_\exists} \mathbf{Val}(\sigma',\tau) - \mathbf{Val}(\sigma,\tau)),$$

let us refer to the witnesses σ and σ' as the *primary strategy* and the *alternative strategy* respectively. Observe that for any primary strategy for Eve and any one strategy for Adam, we can assume Adam plays to maximize the payoff (*i.e.* co-operates) together with the alternative strategy once it deviates (necessarily at an Eve vertex) or to minimize against the primary strategy – again, once it deviates. Indeed, since the deviation yields different histories, the two strategies for Adam can be combined without conflict. More formally,

▶ **Lemma 6.** *Consider any $\sigma \in \mathfrak{S}_\exists$, $\tau \in \mathfrak{S}_\forall$, and corresponding play $\pi_{\sigma\tau} = v_0v_1\dots$. For all $i \geq 0$ such that $v_i \in V_\exists$, for all $v' \in \mathbf{succ}(v_i) \setminus \{v_{i+1}\}$ there exist $\sigma' \in \mathfrak{S}_\exists$, $\tau' \in \mathfrak{S}_\forall$ for which*

(i) $\pi_{\sigma'\tau}[..i+1] = \pi_{\sigma\tau}[..i]\cdot v'$,
(ii) $\mathbf{Val}(\pi_{\sigma'\tau'}[i+1..]) = \mathbf{cVal}^{v'}(G)$, *and*
(iii) $\pi_{\sigma\tau} = \pi_{\sigma\tau'}$.

Furthermore, from any vertex $v \in V$, Eve has a strategy to ensure a payoff of at least $\mathbf{aVal}^v(G)$ and Adam has a strategy to ensure a payoff of at most $\mathbf{aVal}^v(G)$. Thus, one could further assume that Adam plays to minimize against the primary strategy while maximizing against the alternative one.

▶ **Lemma 7.** *Consider any $\sigma \in \mathfrak{S}_\exists$, $\tau \in \mathfrak{S}_\forall$, and corresponding play $\pi_{\sigma\tau} = v_0v_1\dots$. For all $i \geq 0$ such that $v_i \in V_\exists$, for all $v' \in \mathbf{succ}(v_i) \setminus \{v_{i+1}\}$ there exist $\sigma' \in \mathfrak{S}_\exists$, $\tau' \in \mathfrak{S}_\forall$ for which*

(i) $\pi_{\sigma'\tau}[..i+1] = \pi_{\sigma\tau}[..i]\cdot v' = \pi_{\sigma\tau'}[..i]\cdot v'$,
(ii) $\mathbf{Val}(\pi_{\sigma'\tau'}[i+1..]) = \mathbf{cVal}^{v'}(G)$, *and*
(iii) $\mathbf{Val}(\pi_{\sigma\tau'}[i+1..]) \leq \mathbf{aVal}^{v_{i+1}}(G)$.

Both claims follow from the definitions of strategies for Eve and Adam and from Lemma 4.

In the remaining of this work, we will assume that λ is **not** given as part of the input.

3 Regret against all strategies of Adam

In this section we describe an algorithm to compute the (minimal) regret of a discounted-sum game when there are no restrictions placed on the strategies of Adam. The algorithm can be implemented by an alternating machine guaranteed to halt in polynomial time. We show that the regret *value* of any game is achieved by a strategy for Eve which consists of two strategies, the first choosing edges which lead to the optimal co-operative value, the second choosing edges which ensure the antagonistic value. The switch from the former to the latter is done based on the "local regret" of the vertex (this is formalized in the sequel). The latter allows us to claim NP-membership of the regret threshold problem. The following theorem summarizes the bounds we obtain:

▶ **Theorem 8.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly), playing against all strategies of Adam, is in* NP.

Let us start by formalizing the concept of *local regret*. Given a play or play prefix $\pi = v_0 \dots$ and integer $0 \leq i < |\pi|$ such that $v_i \in V_\exists$, define $\mathbf{locreg}(\pi, i)$ as follows:

$$\begin{cases} \lambda^i \left(\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) - \mathbf{Val}(\pi[i..]) \right) & \text{if } \pi \text{ is a play,} \\ \lambda^i \left(\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) - \mathbf{Val}(\pi[i..j]) \right) - \lambda^j \mathbf{aVal}^{v_j}(G) & \text{if } \pi \text{ is a prefix of length } j+1 > i+1, \\ \lambda^i \left(\mathbf{cVal}^{v_i}(G) - \mathbf{aVal}^{v_i}(G) \right) & \text{if } \pi \text{ is a prefix of length } i+1, \end{cases}$$

where $\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) = \max\{w(v_i, v) + \lambda \mathbf{cVal}^v(G) : (v_i, v) \in E \text{ and } v \neq v_{i+1}\}$. Intuitively, for π a play, $\mathbf{locreg}(\pi, i)$ corresponds to the difference between the value of the best *deviation* from position i and the value of π. For π a play prefix, $\mathbf{locreg}(\pi, i)$ assumes that after position $j = |\pi| - 1$ Eve will play a worst-case optimal strategy.

3.1 Deciding 0-regret

We will now argue that the problem of determining whether Eve has a regret-free strategy can be decided in polynomial time. Furthermore, if no such strategy for Eve exists, we will extract a strategy for Adam which, against any strategy of Eve, ensures non-zero regret. To do so, we will reduce the problem to that of deciding whether Eve wins a safety game. The unsafe edges are determined by a function of the antagonistic and co-operative values of the original game. Critically, the game is played on the same arena as the original regret game.

▶ **Theorem 9.** *Deciding if the regret value is* 0*, playing against all strategies of Adam, is in* PTIME*.*

Proof. We define a partition of the edges leaving vertices from $V_\exists$ into good and bad for Eve. A bad edge is one which witnesses non-zero local regret. We then show that Eve can ensure a regret value of 0 if and only if she has a strategy to avoid ever traversing bad edges. More formally, let us assume a given weighted arena $G = (V, V_\exists, v_I, E, w)$ and a discount factor $\lambda \in (0, 1)$. We define the set of bad edges $\mathcal{B} := \{(u, v) \in E : u \in V_\exists \text{ and } w(u, v) + \lambda \mathbf{aVal}^v(G) < \mathbf{cVal}^u_{\neg v}(G)\}$.

Note that strategies for either player in the newly defined safety game are also strategies for them in the original game (and vice versa as well). We now claim that winning strategies for Adam in the safety game $\hat{G} = (V, V_\exists, v_I, E, \mathcal{B})$ ensure that, regardless of the strategy of Eve, its regret will be strictly positive. The idea behind the claim is that, Adam can force to traverse a bad edge and from there, play adversarially against the primary strategy and co-operatively with an alternative strategy.

▶ **Claim 10.** *If $\tau \in \mathfrak{S}_\forall$ is a winning strategy for Adam in $\hat{G}$, then there exist $\tau' \in \mathfrak{S}_\forall$ and $\sigma' \in \mathfrak{S}_\exists$ such that $\forall \sigma \in \mathfrak{S}_\exists : \mathbf{Val}(\sigma', \tau') - \mathbf{Val}(\sigma, \tau') \geq \lambda^{|V|} \min\{\mathbf{cVal}^u_{\neg v}(G) - w(u, v) - \lambda \mathbf{aVal}^v(G) : (u, v) \in \mathcal{B} \text{ and } u \in V_\exists\} > 0$.*

The claim follows from the definitions and Lemma 7. Conversely, winning strategies for Eve in $\hat{G}$ are actually regret-free.

▶ **Claim 11.** *If $\sigma \in \mathfrak{S}_\exists$ is a winning strategy for Eve in $\hat{G}$, then $\mathbf{reg}^\sigma(G) = 0$.*

Our argument to prove this claim requires we first show that a winning strategy for Eve ensures the antagonistic value of G from v_I.

The desired result then follows from Lemma 3 and from the fact that membership of an edge in $\mathcal{B}$ can be decided by computing **cVal** and a threshold query regarding **aVal**, thus in polynomial time. ◀

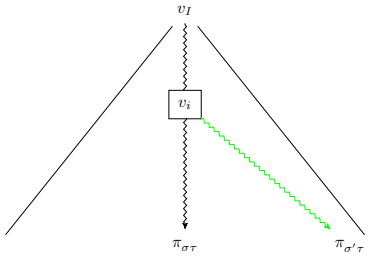


Figure 3 Depiction of a play and a "better alternative play".

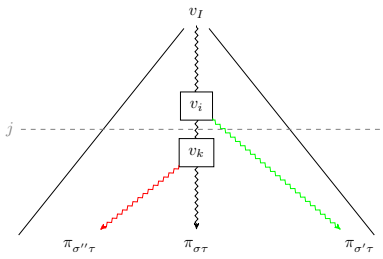


Figure 4 A deviation from v_k cannot be a better alternative to $\pi_{\sigma\tau}$ if $j \geq N(\mathbf{Val}(\sigma', \tau) - \mathbf{Val}(\sigma, \tau))$.

We observe that the proof of Theorem 9 – more precisely, Claim 10 – implies that, if there is no regret-free strategy for Eve in a game, then the regret of the game is at least $\lambda^{|V|}$ times the smallest local regret labelling the bad edge from $\mathcal{B}$ which Adam can force. More formally:

▶ **Corollary 12.** *If no regret-free strategy for Eve exists in G, then $\mathbf{Reg}(G) \geq a_G$ where $a_G := \lambda^{|V|} \min\{\mathbf{locreg}(uv, 0) : u \in V_\exists$ and $(u, v) \in \mathcal{B}\}$.*

3.2 Deciding r-regret

It will be useful in the sequel to define the *regret of a play* and the *regret of a play prefix*. Given a play $\pi = v_0 v_1 \dots$, we define the regret of π as:

$$\mathbf{reg}(\pi) := \sup\left(\{\mathbf{locreg}(\pi, i) : v_i \in V_\exists\} \cup \{0\}\right).$$

Intuitively, the local regrets give lower bounds for the overall regret of a play. We will also let the *regret of a play prefix* $\rho = v_0 \dots v_j$ be equal to

$$\max\left(\{\lambda^i(\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) - \mathbf{Val}(\rho[i..j])) : 0 \leq i < j \text{ and } v_i \in V_\exists\} \cup \{0\}\right).$$

Let us give some more intuition regarding the regret of a play. Consider a pair of strategies σ and τ for Eve and Adam, respectively. Suppose there is an alternative strategy σ' for Eve, such that, against τ, the obtained payoff is greater than that of $\pi_{\sigma\tau}$. It should be clear that this implies there is some position i such that, from vertex $v_i \in V_\exists$ σ' and τ result in a different play from $\pi_{\sigma\tau}$ (see Figure 3). We will sometimes refer to this deviation, *i.e.* the play $\pi_{\sigma'\tau}$, as a *better alternative* to $\pi_{\sigma\tau}$.

We can now show the regret of a strategy for Eve in fact corresponds to the supremum of the regret of plays consistent with the strategy.

▶ **Lemma 13.** *For any strategy σ of Eve, $\mathbf{reg}^\sigma(G) = \sup\{\mathbf{reg}(\pi) : \pi$ is consistent with $\sigma\}$.*

We note that for any play π, the sequence $\langle \lambda^i(\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) - \mathbf{Val}(\pi[i..]))\rangle_{i \geq 0}$ converges to 0 because $(\mathbf{cVal}^{v_i}_{\neg v_{i+1}}(G) - \mathbf{Val}(\pi[i..]))$ is bounded by $\frac{2W}{(1-\lambda)}$. It follows that if we have a non-zero lower bound for the regret of π, then there is some index N such that the witness for the regret occurs before N. Moreover, we can place a polynomial upper bound on N. More precisely:

▶ **Lemma 14.** *Let π be a play in G and suppose $0 < r \leq \mathbf{reg}(\pi)$. Let*

$$N(r) := \lfloor (\log r + \log(1 - \lambda) - \log(2W)) / \log \lambda \rfloor + 1.$$

Then $\mathbf{reg}(\pi) = \mathbf{reg}(\pi[..N(r)]) - \lambda^{N(r)} \mathbf{Val}(\pi[N(r)..])$.

The above result gives us a bound on how far we have to unfold a game after having witnessed a non-zero lower bound, r, for the regret. If we consider the example from Figure 3, this translates into a bound on how many turns after v_i a deviation can still yield bigger local regret (see Figure 4).

Corollary 12 then gives us the required lower bound to be able to use Lemma 14.

▶ **Lemma 15.** *If* $\mathbf{Reg}(G) \geq a_G$ *then* $\mathbf{Reg}(G)$ *is equal to*

$$\inf_{\sigma \in \mathfrak{S}_\exists} \sup\{\mathbf{reg}(\pi[..N(a_G)]) - \lambda^{N(a_G)}\mathbf{aVal}^{v_{N(a_G)}}(G) : \pi = v_0 v_1 \dots \text{ is consistent with } \sigma\}.$$

This already implies we can compute the regret value in alternating polynomial time (or equivalently, deterministic polynomial space [5]).

▶ **Proposition 16.** *The regret value is computable using only polynomial space.*

Proof. We first label the arena with the antagonistic and co-operative values and solve the safety game described for Theorem 9. The latter can be done in polynomial time. If the Eve wins the safety game, the regret value is 0. Otherwise, we know $a_G > 0$ is a lower bound for the regret value. We now simulate G using an alternating Turing machine which halts in at most $N(a_G)$ steps. That is, a polynomial number of steps. The simulated play prefix is then assigned a regret value as per Lemma 15 (recall we have already pre-computed the antagonistic value of every vertex). ◀

As a side-product of the algorithm described in the above proof we get that finite memory strategies suffice for Eve to minimize her regret in a discounted-sum game.

▶ **Corollary 17.** *Let* $\mu := |\Delta|^{N(a_G)}$*, with* $N(0) = 0$*. It holds that*

$$\mathbf{Reg}_{\Sigma^\mu_\exists, \mathfrak{S}_\forall}(G) = \mathbf{Reg}_{\mathfrak{S}_\exists, \mathfrak{S}_\forall}(G).$$

3.3 Simple regret-minimizing behaviours

We will now argue that Eve has a simple strategy which ensures regret of at most $\mathbf{Reg}(G)$. Her strategy will consist of "playing co-operatively" for some turns (until a high local regret has already been witnessed) and then switch to a co-operative worst-case optimal strategy.

We will now define a family of strategies which switch from co-operative behaviour to antagonistic, after a specific number of turns have elapsed (in fact, enough for the discounted local regret to be less than the desired regret). Denote by σ^{co} a strongly co-operative strategy for Eve in G and by σ^{cw} a co-operative worst-case optimal strategy for Eve in G. Recall that, by Lemma 5, such strategies for her always exist. Finally, given a co-operative strategy σ^{co}, a co-operative worst-case optimal strategy σ^{cw}, and $t \in \mathbb{Q}$ let us define an *optimistic-then-pessimistic strategy for Eve* $[\sigma^{\mathsf{co}} \xrightarrow{\mathrm{t}} \sigma^{\mathsf{cw}}]$. The strategy is such that, for any play prefix $\rho = v_0 \dots v_n$ such that $v_n \in V_\exists$

$$[\sigma^{\mathsf{co}} \xrightarrow{\mathrm{t}} \sigma^{\mathsf{cw}}](\rho) = \begin{cases} \sigma^{\mathsf{co}}(\rho) & \text{if } |\mathbf{cOpt}(v_n)| = 1 \text{ and } \mathbf{locreg}(\rho \cdot \sigma^{\mathsf{cw}}(\rho), n+1) > t \\ \sigma^{\mathsf{cw}}(\rho) & \text{otherwise.} \end{cases}$$

We claim that, when we set $t = \mathbf{Reg}(G)$, an optimistic-then-pessimistic strategy for Eve ensures minimal regret. That is

▶ **Proposition 18.** *Let* σ^{co} *be a strongly co-operative strategy for Eve,* σ^{cw} *be a Eve and a co-operative worst-case optimal strategy for Eve, and* $t = \mathbf{Reg}(G)$*. The strategy* $\sigma = [\sigma^{\mathsf{co}} \xrightarrow{\mathrm{t}} \sigma^{\mathsf{cw}}]$ *has the property that* $\mathbf{reg}^\sigma(G) = \mathbf{Reg}(G)$*.*

This is a refinement of the strategy one can obtain from applying the algorithm used to prove Proposition 16.[1] The latter tells us that a regret-minimizing strategy of Eve eventually switches to a worst-case optimal behaviour. For vertices where, before this switch, another edge was chosen by Eve, we argue that she must have been playing a co-operative strategy. Otherwise, she could have switched sooner.

We have shown the regret value can be computed using an algorithm which requires polynomial space only. This algorithm is based on a polynomial-length unfolding of the game and from it we can deduce that the regret value is representable using a polynomial number of bits. (Indeed, all exponents occurring in the formula from Lemma 15 will be polynomial according to Lemma 14.) Also, we have argued that Eve has a "simple" strategy σ to ensure minimal regret. Such a strategy is defined by two polynomial-time constructible sub-strategies and the regret value of the game. Hence, it can be encoded into a polynomial number of bits itself. Furthermore, σ is guaranteed to be playing as its co-operative worst-case optimal component after $N(\mathbf{Reg}(G))$ turns (see, again, Lemma 14), which is a polynomial number of turns. Given a regret threshold r, we claim we can verify that σ ensures regret at most r in polynomial time. This can be achieved by allowing Adam to play in G, and against σ, with the objective of reaching an edge with high local regret before $N(\mathbf{Reg}(G))$ turns. A possible formalization of this idea follows. Consider the product of G with a counter ranging from 1 to $N(\mathbf{Reg}(G))$ where we make all vertices belong to Adam. In this game H, we make edges leaving vertices previously belonging to Eve go to a sink and define a new weight function w' which assigns to these edges their negative non-discounted local regret: going from u to v when σ dictates to go to v' yields $w(u,v')+\lambda\mathbf{aVal}^{v'}(H\times\sigma)-w(u,v)+\lambda\mathbf{cVal}^{v}(H)$. Lemma 15 allows us to show that σ ensures regret at most r in G if and only if the antagonistic value of a discounted-sum game played on H with weight function w' is at most $-r$.

It follows that the regret threshold problem is in NP, as stated in Theorem 8.

▶ **Example 19.** We revisit the discounted-sum game from Figure 1. Let us instantiate the values $M = 100$ and $\lambda = \frac{9}{10}$. According to our previous remarks on this arena, after i visits to v without Adam choosing (v, y), Eve could achieve $\left(\frac{1}{1-\frac{9}{10}}\right)\left(\frac{9}{10}\right)^{2i} = 10\left(\frac{9}{10}\right)^{2i}$, by going to x, or hope for $\left(\frac{100}{1-\frac{9}{10}}\right)\left(\frac{9}{10}\right)^{2i+1} = 1000\left(\frac{9}{10}\right)^{2i+1}$ by going to v again. Her best regret minimizing strategy corresponds to σ^{22} which ensures regret of at most $10 - 10(\frac{9}{10})^{44} \approx 9.9030$. (Note that, since $1000(\frac{9}{10})^{2j+1} < 10$ for all $j \geq 22$, the best alternative strategy indeed achieves a payoff of 10 only. This alternative strategy corresponds to having gone to x from the beginning.)

It is easy to see that Eve cannot win the safety game $\hat{G}$ constructed from this arena. Indeed, from the initial vertex v_I both (v_I, x) and (v_I, v) are bad edges for Eve since there are always alternative strategies to obtain higher payoffs (that is, if Adam does not play to y). More formally, we note that $\mathbf{cVal}^{v_I}_{\neg x} = 1000\lambda$, $\mathbf{cVal}^{v_I}_{\neg v} = 10$, $\mathbf{aVal}^{x} = 10$, and $\mathbf{aVal}^{v} = 10\lambda^2$. Thus, the lower bound a_G one can obtain from $\hat{G}$ is then equal to

$$\min\left\{1000\left(\frac{9}{10}\right) - 10, 10 - 10\left(\frac{9}{10}\right)^2\right\}\left(\frac{9}{10}\right)^4 = 10\left(\frac{9}{10}\right)^4 - 10\left(\frac{9}{10}\right)^6 \approx 1.2466.$$

As expected, when Eve plays her optimal regret-minimizing (optimistic-then-pessimistic) strategy any better alternative must deviate before $N(a_G) = 71$ turns. In general, against σ^i, for $i < 22$ a regret bigger than 9.9030 is obtained by Adam choosing the edge (v, y) to help

[1] In fact, our proof of Prop. 18 relies on Eve requiring finite memory, to minimize her regret.

any strategy of Eve going to v more than i times, for $i \geq 22$ choices of Adam are no longer relevant and the best alternative strategy for Eve is to have gone to x from the first step. ◀

4 Regret against positional strategies of Adam

In this section we consider the problem of computing the (minimal) regret when Adam is restricted to playing positional strategies.

▶ **Theorem 20.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly), playing against positional strategies of Adam, is in* PSPACE.

Playing against an Adam, when he is restricted to playing memoryless strategies gives Eve the opportunity to learn some of Adam's strategic choices. However, due to its decaying nature, with the discounted-sum payoff function Eve must find a balance between exploring too quickly, thereby presenting lightly discounted alternatives; and learning too slowly, thereby heavily discounting her eventual payoff.

A similar approach to the one we have adopted in Section 3 can be used to obtain an algorithm for this setting. The claimed lower bound follows from Theorem 26.

4.1 Deciding 0-regret

As in the previous section, we will reduce the problem of deciding if the game has regret value 0 to that of determining the winner of a safety game. It will be obvious that if no regret-free strategy for Eve exists in the original game, then we can construct, for any strategy of hers, a positional strategy of Adam which ensures non-zero regret. Hence, we will also obtain a lower bound on the regret of the game in the case Adam wins the safety game.

Let us fix some notation. For a set of edges $D \subseteq E$, we denote by $G\lfloor D$ the weighted arena $(V, V_\exists, v_I, D, w)$. Also, for a positional strategy $\tau : (V \setminus V_\exists) \to E$ for Adam in G, we denote by $G \times \tau$ the weighted arena resulting from removing all edges not consistent with τ. Next, for an edge $(s,t) \in E$ we define $E_\forall(st) := \{(u,v) \in E : \text{if } u = s \text{ then } v = t \text{ or } u \in V_\exists\}$. We extend the latter to play prefixes $\rho = v_0 \dots v_n$ by (recursively) defining $E_\forall(\rho) := E_\forall(\rho[..n-1]) \cap E_\forall(v_{n-1}v_n)$. If π is a play, then $E \supseteq E_\forall(\pi[..i]) \supseteq E_\forall(\pi[..j])$ for all $0 \leq i \leq j$. Hence, since E is finite, the value $E_\forall(\pi) := \lim_{i \geq 0} E_\forall(\pi[..i])$ is well-defined. Remark that $E_\forall(\pi)$ does not restrict edges leaving vertices of Eve. The following properties directly follow from our definitions.

▶ **Lemma 21.** *Let π be a play or play prefix consistent with a positional strategy for Adam. It then holds that:*

(i) *for every $v \in V \setminus V_\exists$ there is some edge $(v, \cdot) \in E_\forall(\pi)$,*
(ii) *π is consistent with a strategy $\tau \in \Sigma^1_\forall(G)$ if and only if $\tau \in \Sigma^1_\forall(G\lfloor E_\forall(\pi))$, and*
(iii) *every strategy $\tau \in \Sigma^1_\forall(G\lfloor E_\forall(\pi))$ is also an element from $\Sigma^1_\forall(G)$.*

To be able to decide whether regret-free strategies for Eve exist, we define a new safety game. The arena we consider is $\hat{G} := (\hat{V}, \hat{V_\exists}, \hat{v_I}, \hat{E})$ where $\hat{V} := V \times \mathcal{P}(E)$, $\hat{V_\exists} := V_\exists \times \mathcal{P}(E)$, $\hat{v_I} := (v_I, E)$, and $\hat{E}$ contains the edge $((u,C),(v,D))$ if and only if $(u,v) \in E$ and $D = C \cap E_\forall(uv)$.

▶ **Theorem 22.** *Deciding if the regret value is* 0*, playing against positional strategies of Adam, is in* PSPACE.

Proof. A safety game is constructed as in the proof of Theorem 9. Here, we consider $\tilde{G}$ and the set of bad edges $\tilde{\mathcal{B}} := \{((u,C),(v,D)) \in \hat{E} : u \in V_\exists \text{ and } \exists \tau \in \Sigma^1_\forall(G\lfloor C), w(u,v) +$

$\lambda\mathbf{cVal}^v(G \times \tau) < \mathbf{cVal}^u_{\neg v}(G \times \tau)\}$. We then have the safety game $\tilde{G} = (\hat{V}, \hat{V}_\exists, \hat{v_I}, \hat{E}, \tilde{\mathcal{B}})$. Note that there is an obvious bijective mapping from plays (and play prefixes) in $\tilde{G}$ to plays (prefixes) in G which are consistent with a positional strategy for Adam. One can then show the following properties hold:

▶ **Claim 23.** *If $\tau \in \mathfrak{S}_\forall(\tilde{G})$ is a winning strategy for Adam in $\tilde{G}$, then for all $\sigma \in \mathfrak{S}_\exists(G)$, there exist $t_{\tau\sigma} \in \Sigma^1_\forall(G)$ and $s_{\tau\sigma} \in \mathfrak{S}_\exists(G)$ such that $\mathbf{Val}(s_{\tau\sigma}, t_{\tau\sigma}) - \mathbf{Val}(\sigma, t_{\tau\sigma}) \geq \lambda^{|V|(|E|+1)}$ $\min\{\mathbf{cVal}^u_{\neg v}(G \times \tau) - w(u,v) - \lambda\mathbf{cVal}^v(G \times \tau) : ((u,C),(v,D)) \in \tilde{\mathcal{B}}, \tau \in \Sigma^1_\forall(G\lfloor C)\}$.*

The claim follows from *positional determinacy* of safety games and Lemma 21.

▶ **Claim 24.** *If $\sigma \in \mathfrak{S}_\exists(\tilde{G})$ is a winning strategy for Eve in $\tilde{G}$, then there is $s_\sigma \in \mathfrak{S}_\exists(G)$ such that $\mathbf{reg}^{s_\sigma}_{\mathfrak{S}_\exists, \Sigma^1_\forall}(G) = 0$.*

It then follows from the determinacy of safety games that Eve wins the safety game $\tilde{G}$ if and only if she has a regret-free strategy.

We observe that simple cycles in $\tilde{G}$ have length at most $|V|(|E|+1)$. Thus, we can simulate the safety game until we complete a cycle and check that all traversed edges are good, all in alternating polynomial time. Indeed, an alternating Turing machine can simulate the cycle and then (universally) check that for all edges, for all positional strategies of the Adam, the inequality holds. ◀

▶ **Corollary 25.** *If no regret-free strategy for Eve exists in G, then $\mathbf{Reg}_{\mathfrak{S}_\exists, \Sigma^1_\forall}(G) \geq b_G$ where $b_G := \lambda^{|V|(|E|+1)} \min\{\mathbf{cVal}^u_{\neg v}(G \times \tau) - w(u,v) - \lambda\mathbf{cVal}^v(G \times \tau) : ((u,C),(v,D)) \in \tilde{\mathcal{B}}$ and $\tau \in \Sigma^1_\forall(G\lfloor C)\}$.*

4.2 Lower bounds

We claim that both 0-regret and r-regret are coNP-hard. This can be shown by adapting the reduction from 2-disjoint-paths given in [13] to the regret threshold problem against memoryless adversaries.

▶ **Theorem 26.** *Let $\lambda \in (0,1)$ and $r \in \mathbb{Q}$ be fixed. Deciding if the regret value is less than r (strictly or non-strictly), playing against positional strategies of Adam, is coNP-hard.*

5 Playing against word strategies of Adam

In this section, we consider the case where Adam is restricted to playing *word strategies*. First, we show that the regret threshold problem can be solved whenever the discounted sum automata associated to the game structure can be made deterministic. As the determinization problem for discounted sum automata has been solved in the literature for only sub-classes of discount factors, and left open in the general case, we complement this result by two other results. First, we show how to solve an *ε-gap promise* variant of the regret threshold problem, and second, we give an algorithm to solve the 0 regret problem. In the two cases, we obtain completeness results on the computational complexities of the problems.

5.1 Preliminaries

The formal definition of the ε-gap promise problem is given below. We first define here the necessary vocabulary. We say that a strategy of Adam is a *word strategy* if his strategy can be expressed as a function $\tau : \mathbb{N} \to [\max\{\mathbf{deg}^+(v) : v \in V\}]$, where $[n] = \{i : 1 \leq i \leq n\}$ and $\mathbf{deg}^+(v)$ is the *outdegree* of v (i.e. the number of edges leaving v). Intuitively, we consider an

order on the successors of each Adam vertex. On the i-th turn, the strategy τ of Adam will tell him to move to the $\tau(i)$-th successor of the vertex according to the fixed order. We denote by $\mathfrak{W}_\forall$ the set of all such strategies for Adam. A game in which Adam plays word strategies can be reformulated as a game played on a weighted automaton $\Gamma = (Q, q_I, A, \Delta, w)$ and strategies of Adam– of the form $\tau : \mathbb{N} \to A$ – determine a sequence of input symbols, i.e. an omega word, to which Eve has to react by choosing Δ-successor states starting from q_I. In this setting a strategy of Eve which minimizes regret defines a run by resolving the non-determinism of Δ in Γ, and ensures the difference of value given by the constructed run is minimal w.r.t. to the value of the best run on the word spelled out by Adam.

5.2 Deciding 0-regret

We will now show that if the regret of an arena (or automaton) is 0, then we can construct a memoryless strategy for Eve which ensures no regret is incurred. More specifically, assuming the regret is 0, we have the existence of a family of strategies of Eve which ensure decreasing regret (with limit 0). We use this fact to choose a small enough ε and the corresponding strategy of hers from the aforementioned family to construct a memoryless strategy for Eve with nice properties which allow us to conclude that its regret is 0. Hence, it follows that an automaton has zero regret if and only if a memoryless strategy of Eve ensures regret 0. As we can guess such a strategy and easily check if it is indeed regret-free (using the obvious reduction to non-emptiness of discounted-sum automata or one-player discounted-sum games), the problem is in NP. A matching lower bound follows from a reduction from SAT which was first described in [1].

▶ **Theorem 27.** *Deciding if the regret value is* 0*, playing against word strategies of Adam, is* NP*-complete.*

5.3 Deciding r-regret: determinizable cases

When the weighted automaton Γ associated to the game structure can be made deterministic, we can solve the regret threshold problem with the following algorithm. In [13] we established that, against eloquent adversaries, computing the regret reduced to computing the value of a quantitative simulation game as defined in [6]. The game is obtained by taking the product of the original automaton and a deterministic version of it. The new weight function is the difference of the weights of both components (for each pair of transitions). In [4], it is shown how to determinize discounted-sum automata when the discount factor is of the form $\frac{1}{n}$, for $n \in \mathbb{N}$. So, for this class of discount factor, we can state the following theorem:

▶ **Theorem 28.** *Deciding if the regret value is less than a given threshold (strictly or non-strictly), playing against word strategies of Adam, is in* EXP *for λ of the form $\frac{1}{n}$.*

5.4 The ε-gap promise problem

Given a discounted-sum automaton $\mathcal{A}$, $r \in \mathbb{Q}$, and $\varepsilon > 0$, the ε-gap promise problem adds to the regret threshold problem the hypothesis that $\mathcal{A}$ will either have regret $\leq r$ or $> r + \varepsilon$. We observe that an algorithm that satisfies that:

- a YES answer implies that $\mathbf{Reg}_{\Sigma_\exists, \mathfrak{W}_\forall}(\mathcal{A}) \leq r + \varepsilon$,
- whereas a NO answer implies $\mathbf{Reg}_{\Sigma_\exists, \mathfrak{W}_\forall}(\mathcal{A}) > r$.

will decide the ε-gap promise problem.

In [4], it is shown that there are discounted-sum automata which define functions that cannot be realized with deterministic-sum automata. Nevertheless, it is also shown in that

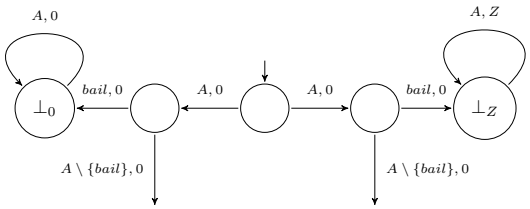


Figure 5 Initial gadget used in reduction from QBF.

paper that given a discounted-sum automaton it is always possible to construct a deterministic one that is ε-close in the following formal sense. A *discounted-sum automaton $\mathcal{A}$ is ε-close to another discounted sum automaton $\mathcal{B}$*, if for all words x the absolute value of the difference between the values assign by $\mathcal{A}$ and $\mathcal{B}$ to x is at most ε. So, it should be clear that we can apply the algorithm underlying Theorem 28 to Γ and a determinized version $\mathcal{D}_\Gamma$ of it (which is ε-close to Γ) and solve the ε-gap promise problem. We can then prove the following result.

▶ **Theorem 29.** *Deciding the ε-gap regret problem is in* PSPACE.

The complexity of the algorithm follows from the fact that the value of a (quantitative simulation) game, played on the product of Γ and $\mathcal{D}_\Gamma$ we described above, can be determined by simulating the game for a polynomial number of turns. Thus, although the automaton constructed using the techniques of Boker and Henzinger [4] is of size exponential, we can construct it "on-the-fly" for the required number of steps and then stop.

Proof Sketch. We reduce the problem to determining the winner of a reachability game on an exponentially larger arena. Although the arena is exponentially larger, all paths are only polynomial in length, so the winner can be determined in alternating polynomial time, or equivalently, polynomial space.

The idea of the construction is as follows. Given a discounted-sum automaton $\mathcal{A}$, we determinize its transitions via a subset construction, to obtain a deterministic, multi-valued discounted-sum automaton $D_\mathcal{A}$. Then we decide if Eve is able to simulate, within the regret bound, the $D_\mathcal{A}$ on $\mathcal{A}$ for all *finite* words up to a length (polynomially) dependent on ε. If we simulate the automaton for a sufficient number of steps, then any significant gap between the automata will be unrecoverable regardless of future inputs, and we can give a satisfactory answer for the ε-GAP REGRET PROBLEM. More specifically, we only have to simulate this determinization process for N steps, where $N := \lfloor(\log\varepsilon + \log(1-\lambda) - \log(4W))/\log\lambda\rfloor + 1$. ◀

5.5 Lower bounds

We claim the ε-gap promise problem is PSPACE-hard even if both λ and ε are not part of the input. To establish the result, we give a reduction from QSAT which uses the gadgets depicted in Figures 5 and 6.

▶ **Theorem 30.** *Let $\lambda \in (0,1)$ and $\varepsilon \in (0,1)$ be fixed. As input, assume we are given $r \in \mathbb{Q}$ and weighted arena $\mathcal{A}$ such that $\mathbf{Reg}_{\Sigma_\exists,\mathfrak{W}_\forall}(\mathcal{A}) \leq r$ or $\mathbf{Reg}_{\Sigma_\exists,\mathfrak{W}_\forall}(\mathcal{A}) > r + \varepsilon$. Deciding if the regret value is less than a given threshold, playing against word strategies of Adam, is* PSPACE*-hard.*

Proof Sketch. We will reduce the QSAT PROBLEM to the present one.

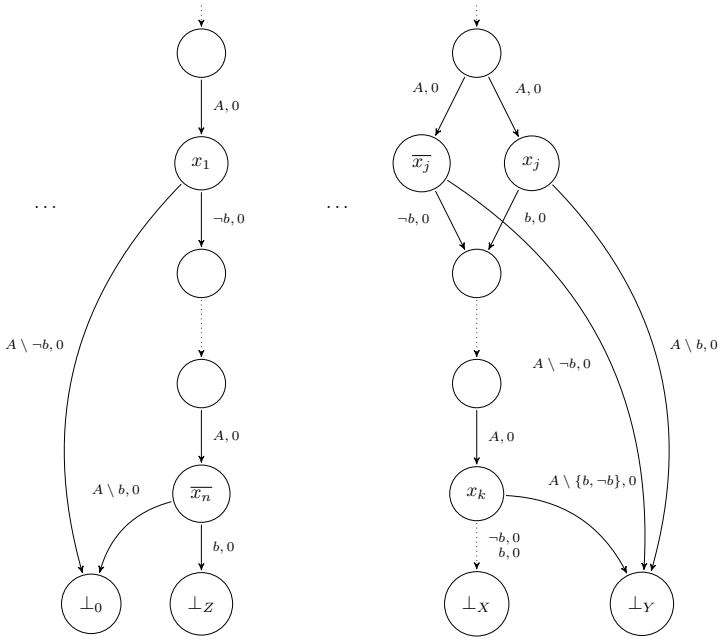


Figure 6 Left and right sub-arenas of the reduction from QBF. Clause i shown on the left; existential and universal gadgets for variables x_j and x_k, respectively, on the right.

The QSAT Problem asks whether a given fully quantified Boolean formula (QBF) is satisfiable. The problem is known to be PSPACE-complete [9]. It is known the result holds even if the formula is assumed to be in conjunctive normal form with three literals per clause (also known as 3-CNF). Therefore, w.l.o.g., we consider an instance of the QSAT Problem to be given in the form:

$$\exists x_0 \forall x_1 \exists x_2 \ldots \Phi(x_0, x_1, \ldots, x_n)$$

where Φ is in 3-CNF.

Our reduction works for values of r, X, Y, and Z such that

(i) $\lambda^2 Z > (r + \varepsilon)(1 - \lambda)$,
(ii) $\lambda^{2n}(Z - X) > (r + \varepsilon)(1 - \lambda)$,
(iii) $\lambda^{2n}(Z - Y) \leq r(1 - \lambda)$,
(iv) $\lambda^3 Y - \lambda^{2n} X \leq r(1 - \lambda)$.

The alphabet of the new weighted arena is $A = \{bail, b, \neg b\}$.

Intuitively, the construction works as follows. The initial gadget shown in Figure 5, ensures that Eve always plays to the rightmost sub-arena. In this part of the game, she has, for every existentially quantified variable, a choice which corresponds to setting the variable to a value of true or false. Depending on her choice, Adam can either play a specific symbol corresponding to him accepting the valuation or stop the simulation and go to $\bot_Y$ where Eve wins the regret game. For universally quantified variables, Eve has no choice (see right part of Fig. 6). The crucial idea is that Adam accepts or rejects a valuation of a variable by playing letter b or $\neg b$. These letters can only be read by a clause gadget if the literal does not make the clause true.

If the QBF is true, Eve will be able to make sure no alternative play in the left side of the game leads to $\bot_Z$. This is the case since the construction ensures that every clause gadget (see left part of Fig. 6) with a literal set to true cannot be used to reach $\bot_Z$. ◀

It follows that the general problem is also PSPACE-hard (even if ε is set to 0).

▶ **Corollary 31.** *Let $\lambda \in (0,1)$. For $r \in \mathbb{Q}$, weighted arena G, determining whether $\mathbf{Reg}_{\mathfrak{S}_\exists,\mathfrak{W}_\forall}(G) \lhd r$, for $\lhd \in \{<,\leq\}$, is* PSPACE*-hard.*

References

1. Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 2010. doi:10.1145/1721837.1721844.
2. Krzysztof R. Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.
3. David E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982. doi:10.1287/opre.30.5.961.
4. Udi Boker and Thomas A. Henzinger. Exact and approximate determinization of discounted-sum automata. *LMCS*, 10(1), 2014. doi:10.2168/LMCS-10(1:10)2014.
5. Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.
6. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4), 2010. doi:10.1145/1805950.1805953.
7. Werner Damm and Bernd Finkbeiner. Does it pay to extend the perimeter of a world model? In *FM*, volume 6664 of *LNCS*, pages 12–26. Springer, 2011. doi:10.1007/978-3-642-21437-0_4.
8. Emmanuel Filiot, Tristan Le Gall, and Jean-François Raskin. Iterated regret minimization in game graphs. In *MFCS*, volume 6281 of *LNCS*, pages 342–354. Springer, 2010.
9. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
10. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006. doi:10.1007/11874683_26.
11. Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Mean-payoff games with partial-observation – (extended abstract). In *RP*, pages 163–175, 2014. doi:10.1007/978-3-319-11439-2_13.
12. Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Minimizing regret in discounted-sum games. *CoRR*, abs/1511.00523, 2015. URL: http://arxiv.org/abs/1511.00523.
13. Paul Hunter, Guillermo A Pérez, and Jean-François Raskin. Reactive synthesis without regret. *Acta Informatica*, pages 1–37, 2015.
14. Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update rules. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1984, Washington, DC, USA*, pages 488–492. ACM, 1984.
15. Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *TCS*, 158(1):343–359, 1996.

Easy to Win, Hard to Master: Optimal Strategies in Parity Games with Costs*

Alexander Weinert[1] and Martin Zimmermann[2]

1 **Reactive Systems Group, Saarland University, Saarbrücken, Germany**
`weinert@react.uni-saarland.de`
2 **Reactive Systems Group, Saarland University, Saarbrücken, Germany**
`zimmermann@react.uni-saarland.de`

Abstract

The winning condition of a parity game with costs requires an arbitrary, but fixed bound on the distance between occurrences of odd colors and the next occurrence of a larger even one. Such games quantitatively extend parity games while retaining most of their attractive properties, i.e, determining the winner is in NP and co-NP and one player has positional winning strategies.

We show that the characteristics of parity games with costs are vastly different when asking for strategies realizing the minimal such bound: the solution problem becomes PSPACE-complete and exponential memory is both necessary in general and always sufficient. Thus, playing parity games with costs optimally is harder than just winning them. Moreover, we show that the tradeoff between the memory size and the realized bound is gradual in general.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Parity Games with Costs, Optimal Strategies, Memory Requirements, Tradeoffs

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.31

1 Introduction

Recently, the focus of research into infinite games for the synthesis of reactive systems moved from studying qualitative winning conditions to quantitative ones. This paradigm shift entails novel research questions, as quantitative conditions induce a (partial) ordering of winning strategies. In particular, there is a notion of semantic optimality for strategies which does not appear in the qualitative setting. Thus, in the quantitative setting, one can ask whether computing optimal strategies is harder than computing arbitrary ones, whether optimal strategies are necessarily larger than arbitrary ones, and whether there are tradeoffs between different quality measures for strategies, e.g., between the size of the strategy and its semantic quality (in terms of satisfaction of the winning condition).

As an introductory example consider the classical (max)-parity condition, which is defined for an infinite sequence drawn from a finite subset of the natural numbers, so-called colors. The parity condition is satisfied if almost all occurrences of an odd color are *answered* by a later occurrence of a larger even color, e.g., the sequence

$$\pi = 1\,0\,2\;1\,0\,0\,2\;1\,0\,0\,0\,2\;1\,0\,0\,0\,0\,2\;1\,0\,0\,0\,0\,0\,2\;1\,0\,0\,0\,0\,0\,0\,2\;1\,0\,0\,0\,0\,0\,0\,0\,2\cdots$$

satisfies the parity condition, as every 1 is eventually answered by a 2.

* Supported by the project "TriCS" (ZI 1516/1-1) of the German Research Foundation (DFG).

© Alexander Weinert and Martin Zimmermann;
licensed under Creative Commons License CC-BY

The finitary parity condition [9] is obtained by additionally requiring the existence of a bound b such that almost every odd color is answered within at most b steps, i.e., π does not satisfy the finitary parity condition, as the length of the zero-blocks is unbounded. Thus, solving a finitary parity game is a boundedness problem: in order to satisfy the condition, an arbitrary, but fixed bound has to be met. In particular, winning strategies for finitary parity games are naturally ordered by the minimal bound they realize along all consistent plays. Thus, finitary parity games induce an optimization problem: compute an optimal winning strategy, i.e., one that guarantees the smallest possible bound.

Other examples for such quantitative winning conditions include mean payoff [11, 30] and energy [3, 5] conditions and their combinations and extensions, request-response conditions [17, 27], finitary parity [9] and parity with costs [15], and parameterized extensions of Linear Temporal Logic (LTL) [1, 13, 19, 28, 29]. Often, these conditions are obtained by interpreting a classical qualitative winning condition quantitatively, e.g., the finitary parity condition.

Often, the best known algorithms for solving such boundedness conditions are as fast as the best ones for their respective qualitative variant, while the fastest known algorithms for the optimization problem are worse. For example, solving games with winning conditions in Prompt-LTL, a quantitative variant of LTL, is 2ExpTime-complete [19] (i.e., as hard as solving classical LTL games [23]), while computing optimal strategies is only known to be in 3ExpTime [28]. The same is true for the sizes of strategies, which jumps from tight doubly-exponential bounds to triply-exponential upper bounds. The situation is similar for other winning conditions as well, e.g., request-response conditions [17]. These examples all have in common that there are no known lower bounds on the complexity and the memory requirements in the optimization variant, except for the trivial ones for the qualitative case. A notable exception are finitary parity games, which are solvable in polynomial time [9] and thus simpler than parity games (according to the state-of-the-art).

In this work, we study optimal strategies in parity games with costs, a generalization of finitary parity games. In this setting, we are able to show that computing optimal strategies is indeed harder than computing arbitrary strategies, and that optimal strategies have exponentially larger memory requirements in general. A parity game with costs is played in a finite directed graph whose vertices are partitioned into the positions of Player 0 and the positions of Player 1. Starting at an initial vertex, the players move a token through the graph: if it is placed at a vertex of Player i, then this player has to move it to some successor. Thus, after ω rounds, the players have produced an infinite path through the graph, a so-called play. The vertices of the graph are colored by natural numbers and the edges are labeled by a cost (encoded in unary). These two labelings induce the parity condition with costs: there has to be a bound b such that almost all odd colors are followed by a larger even color such that the cost between these two positions is at most b. Thus, the sequence π from above satisfies the parity condition with costs, if the cost of the zero-blocks is bounded. Note that the finitary parity condition is the special case where every edge has cost one and the parity condition is the special case where every edge has cost zero.

Thus, to win a parity game with costs, Player 0 has to bound the cost between requests and their responses along all plays. If Player 0 has any such strategy, then she has a positional strategy [15], i.e., a strategy that determines the next move based only on the vertex the token is currently at, oblivious to the history of the play. If we let n denote the number of vertices of the graph the game is played in, then such a strategy uniformly bounds the costs to some bound $b \leq n$ [15], which we refer to as the cost of the strategy. Furthermore, Mogavero et al. showed that the winner of a parity game with costs can be determined in UP $\cap$ co-UP [21]. All previous work on parity games with costs was concerned with the

boundedness variant, i.e., the problems ask to find some bound, not necessarily the best one. Here, in contrast, we study optimal strategies in parity games with costs.

1.1 Our Contribution

Our first result shows that determining whether Player 0 has a strategy whose cost is smaller than a given bound b is PSpace-complete. Thus, computing the bound of an optimal strategy is strictly harder than just deciding whether or not some bound exists (unless PSpace $\subseteq$ UP $\cap$ co-UP). The hardness result is shown by a reduction from QBF and uses the bound b to require Player 0's strategy to implement a satisfying Skolem function for the formula, where picking truth values is encoded by requests of odd colors. The lower bound is complemented by a polynomial space algorithm that is obtained from an alternating polynomial time Turing machine that simulates a finite-duration variant of parity games with costs that is won by Player 0 if and only if she can enforce a cost of at most b in the original game. To obtain the necessary polynomial upper bound on the play length we rely on the upper bound n on the optimal bound and on pumping arguments to deal with the costs along the play, and on a first-cycle variant of parity games [2] to capture the parity condition in parts of the graph where all edges have cost zero.

Our second result concerns memory requirements of optimal strategies. A corollary of the correctness of the finite-duration game yields exponential upper bounds: if Player 0 has a strategy of cost b, then also one of cost b and of size $(b+2)^d \cdot (n+1) = 2^{d \log(b+2)} \cdot (n+1)$, where d is the number of odd colors in the game.

As a third result, we show that this bound is in general tight: we present a family $\mathcal{G}_d$ of parity games with costs such that $\mathcal{G}_d$ has d odd colors and Player 0 requires strategies of size $2^d - 2$ to play optimally in each $\mathcal{G}_d$. This result is based on using the bound b to require Player 0 to store which odd colors have an open request and in which order they were posed. Our result improves a linear bound presented by Chatterjee and Fijalkow [8].

Finally, we study the tradeoff between memory size and cost of a strategy witnessed by the results above: arbitrary winning strategies are as small as possible, i.e., positional, but in general have cost n. In contrast, optimal strategies realize a smaller bound, but might have exponential size. Hence, one can trade cost for memory and vice versa.

Our fourth result shows that this tradeoff is gradual in the games $\mathcal{G}_d$: there are strategies $\sigma_1, \sigma_2, \ldots, \sigma_d$ such that $1 = |\sigma_1| < |\sigma_2| < \cdots < |\sigma_d| = 2^d - 2$ and $b_1 > b_2 > \cdots > b_d$, where b_j is the cost of σ_j. Furthermore, we show that the strategy σ_j has minimal size among all strategies of cost b_j. Equivalently, the strategy σ_j has minimal cost among all strategies whose size is not larger than σ_j's size.

Both lower bounds we prove and the tradeoff result already hold for the special case of finitary parity games, which can even be solved in polynomial time [9]. Hence, in this case, the gap between just winning and playing optimally is even larger. Also, our results are straightforwardly extendable to both bounded variants, i.e., bounded parity games [9] and bounded parity games with costs [15].

All proofs omitted due to space restrictions can be found in the full version [25].

1.2 Related Work

Tradeoffs in infinite games have been studied before, e.g., in stochastic and timed games, one can trade memory for randomness, i.e., randomized strategies are smaller than deterministic ones [7, 10]. A detailed overview of more recent results in this direction and of tradeoffs in

multi-dimensional winning conditions is given in the thesis of Randour [24]. The nature of these results is quite different from ours.

Lang investigated optimal strategies in the resource reachability problem on pushdown graphs [20], where there exist a finite number of counters, which may be increased and reset, but not read during a play. He shows that in order to keep the values of the counters minimal during the play, exponential memory in the number of counters is both necessary and sufficient for Player 0. While the author shows the corresponding decision problem to be decidable, he does not provide a complexity analysis of the problem. Furthermore, the setting of the problem is quite different to the model considered in this work: he considers infinite graphs and multiple counters, but only reachability conditions, while we consider finite graphs and implicit counters tied to the acceptance condition, which is a general parity condition.

Also, Fijalkow et al. proved the non-existence of a certain tradeoff between size and quality of strategies in boundedness games [14], which refuted a conjecture with important implications for automata theory and logics. Such games are similar to those considered by Lang in that they are played in potentially infinite arenas and have multiple counters.

Finally, our results have been recently rephrased in terms of window-parity games [4], another quantitative variant of parity games.

2 Preliminaries

We denote the non-negative integers by $\mathbb{N}$ and define $[n] = \{0, 1, \dots, n-1\}$ for every $n \geq 1$.

An *arena* $\mathcal{A} = (V, V_0, V_1, E, v_I)$ consists of a finite, directed graph (V, E), a partition $\{V_0, V_1\}$ of V into the positions of Player 0 (drawn as circles) and Player 1 (drawn as rectangles), and an initial vertex $v_I \in V$. A *play* in $\mathcal{A}$ is an infinite path $\rho = v_0 v_1 v_2 \cdots$ through (V, E) starting in v_I. To rule out finite plays, we require every vertex to be non-terminal. A *game* $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ with vertex set V and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0. The set of winning plays for Player 1 is $V^\omega \setminus \mathrm{Win}$.

A *strategy* for Player i is a mapping $\sigma \colon V^* V_i \to V$ where $(v, \sigma(wv)) \in E$ for all $wv \in V^* V_i$. We say that σ is *positional* if $\sigma(wv) = \sigma(v)$ for every $wv \in V^* V_i$. We often view positional strategies as a mapping $\sigma \colon V_i \to V$. A play $v_0 v_1 v_2 \cdots$ is *consistent* with a strategy σ for Player i, if $v_{j+1} = \sigma(v_0 \cdots v_j)$ for every j with $v_j \in V_i$. A strategy σ for Player i is a *winning strategy* for $\mathcal{G}$ if every play that is consistent with σ is won by Player i. If Player i has a winning strategy, then we say she wins $\mathcal{G}$. *Solving* a game amounts to determining its winner.

A *memory structure* $\mathcal{M} = (M, m_I, \mathrm{Upd})$ for an arena (V, V_0, V_1, E, v_I) consists of a finite set M of memory states, an initial memory state $m_I \in M$, and an update function $\mathrm{Upd} \colon M \times E \to M$. The update function can be extended to finite play prefixes in the usual way: $\mathrm{Upd}^+(m, v) = m$ and $\mathrm{Upd}^+(m, wvv') = \mathrm{Upd}(\mathrm{Upd}^+(m, wv), (v, v'))$ for $w \in V^*$ and $(v, v') \in E$. A next-move function for Player i $\mathrm{Nxt} \colon V_i \times M \to V$ has to satisfy $(v, \mathrm{Nxt}(v, m)) \in E$ for all $v \in V_i$ and all $m \in M$. It induces a strategy σ for Player i with memory $\mathcal{M}$ via $\sigma(v_0 \cdots v_n) = \mathrm{Nxt}(v_n, \mathrm{Upd}^+(m_I, v_0 \cdots v_n))$. A strategy is called *finite-state* if it can be implemented by a memory structure. We define $|\mathcal{M}| = |M|$. The size of a finite-state strategy is the size of a smallest memory structure implementing it.

An arena $\mathcal{A} = (V, V_0, V_1, E, v_I)$ and a memory structure $\mathcal{M} = (M, m_I, \mathrm{Upd})$ for $\mathcal{A}$ induce the expanded arena $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E', (v_I, m_I))$ where $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{Upd}(m, (v, v')) = m'$. Every play $\rho = v_0 v_1 v_2 \cdots$ in $\mathcal{A}$ has a unique extended play $\mathrm{ext}(\rho) = (v_0, m_0)(v_1, m_1)(v_2, m_2) \cdots$ in $\mathcal{A} \times \mathcal{M}$ defined by $m_0 = m_I$ and $m_{n+1} = \mathrm{Upd}(m_n, (v_n, v_{n+1}))$, i.e., $m_n = \mathrm{Upd}^+(m_I, v_0 \cdots v_n)$. The extended play of a finite play prefix in $\mathcal{A}$ is defined similarly.

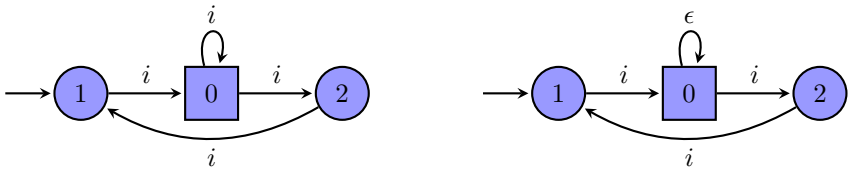


Figure 1 Two parity games with costs. Player 1 only has a winning strategy in the left game.

3 Parity Games with Costs

In this section, we introduce the parity condition with costs [15]. Fix an arena $\mathcal{A} = (V, V_0, V_1, E, v_I)$. A cost function for $\mathcal{A}$ is an edge-labeling $\mathrm{Cst}\colon E \to \{\epsilon, i\}$. [1] Edges labeled with i are called increment-edges while edges labeled with ϵ are called ϵ-edges. We extend the edge-labeling to a cost function over plays obtained by counting the number of increment-edges traversed during the play, i.e., $\mathrm{Cst}(\rho) \in \mathbb{N} \cup \{\infty\}$. The cost of a finite path is defined analogously. Also, fix a coloring $\Omega\colon V \to \mathbb{N}$ of $\mathcal{A}$'s vertices and let $\mathrm{Ans}(c) = \{c' \in \mathbb{N} \mid c' \geq c \text{ and } c' \text{ is even}\}$ be the set of colors that answer a request of color c.

Let $\rho = v_0 v_1 v_2 \cdots$ be a play. We define the cost-of-response at position $k \in \mathbb{N}$ of ρ by

$$\mathrm{Cor}(\rho, k) = \min\{\mathrm{Cst}(v_k \cdots v_{k'}) \mid k' \geq k \text{ and } \Omega(v_{k'}) \in \mathrm{Ans}(\Omega(v_k))\} \ ,$$

where we use $\min \emptyset = \infty$, i.e., $\mathrm{Cor}(\rho, k)$ is the cost of the infix of ρ from position k to its first answer, and ∞ if there is no answer.

We say that a request at position k is answered with cost b, if $\mathrm{Cor}(\rho, k) = b$. Consequently, a request at a position k with an even color is answered with cost zero. Furthermore, we say that a request at position k is unanswered with cost ∞, if there is no position $k' \geq k$ such that $\Omega(v_{k'}) \in \mathrm{Ans}(\Omega(v_k))$ and we have $\mathrm{Cst}(v_k v_{k+1} v_{k+2} \cdots) = \infty$, i.e., there are infinitely many increment-edges after position k, but no answer. There is a third alternative: a request can be unanswered with finite cost, i.e., in case it is not answered, but the play ρ contains only finitely many increment-edges. Still, the cost-of-response is infinite in this case.

The parity condition with costs is defined as

$$\mathrm{CostParity}(\Omega, \mathrm{Cst}) = \{\rho \in V^\omega \mid \limsup_{k \to \infty} \mathrm{Cor}(\rho, k) < \infty\} \ ,$$

i.e., ρ satisfies the condition, if there exists a bound $b \in \mathbb{N}$ such that all but finitely many requests are answered with cost less than b. In particular, only finitely many requests may be unanswered, even with finite cost. Note that the bound b depends on the play ρ.

A game $\mathcal{G} = (\mathcal{A}, \mathrm{CostParity}(\Omega, \mathrm{Cst}))$ is called a parity game with costs. If Cst assigns ϵ to every edge, then $\mathrm{CostParity}(\Omega, \mathrm{Cst})$ is a classical (max-) parity condition, denoted by $\mathrm{Parity}(\Omega)$. Dually, if Cst assigns i to every edge, then $\mathrm{CostParity}(\Omega, \mathrm{Cst})$ is equal to the finitary parity condition over Ω, as introduced by Chatterjee et al. [9] and denoted by $\mathrm{FinParity}(\Omega)$. In these cases, we refer to $\mathcal{G}$ as a parity or a finitary parity game, respectively.

Player 1 has two ways of winning a parity game with costs: Either he violates the classical parity condition, or he delays answers to requests arbitrarily. Consider the two parity games with costs shown in Figure 1. In the game on the left-hand side, Player 1 has a winning strategy, by taking the self-loop of the node labeled with 0 n times upon the n-th visit to it.

[1] Note that using the abstract costs ϵ and i essentially entails a unary encoding of weights. We discuss the case of a binary encoding of arbitrary weights in Section 7.

Thus, he delays answers to the request for 1 arbitrarily and wins by the second condition. In the game on the right-hand side, however, Player 1 does not have a winning strategy. If he eventually remains in the node labeled with 0, then there are only finitely many requests, only one of which is unanswered. Thus, the cost of the play is 0, i.e., it is won by Player 0. If he, on the other hand, always leaves the node labeled with 0 eventually, then each request is answered with cost 2, hence Player 0 wins as well.

▶ **Theorem 1.**
1. *Solving parity games is in* UP ∩ CO-UP. *The winner has a positional winning strategy [12, 18, 22].*
2. *Solving finitary parity games is in* PTIME. *If Player* 0 *wins, then she has a positional winning strategy, but Player* 1 *has in general no finite-state winning strategy [9].*
3. *Solving parity games with costs is in* UP ∩ CO-UP. *If Player* 0 *wins, then she has a positional winning strategy, but Player* 1 *has in general no finite-state winning strategy [15].*

A winning strategy for Player 0 in a parity game with costs does not have to realize a uniform bound b on the value $\limsup_{k\to\infty} \mathrm{Cor}(\rho, k)$ among all plays ρ that are consistent with σ, but the bound may depend on the play. To capture the cost of a strategy, we first define the cost of a play ρ as $\mathrm{Cst}(\rho) = \limsup_{k\to\infty} \mathrm{Cor}(\rho, k)$ and the cost of a strategy σ as $\mathrm{Cst}(\sigma) = \sup_\rho \mathrm{Cst}(\rho)$, where the supremum ranges over all plays ρ that are consistent with σ. A strategy is optimal for $\mathcal{G}$ if it has minimal cost among all strategies for $\mathcal{G}$.

A corollary of Theorem 1.3 yields an upper bound on the cost of an optimal strategy: a straightforward pumping argument shows that a positional winning strategy, which always exists if there exists any winning strategy, realizes a uniform bound $b \leq n$ for every play, where n is the number of vertices of the game [15].

▶ **Corollary 2.** *Let $\mathcal{G}$ be a parity game with costs with n vertices. If Player* 0 *wins $\mathcal{G}$, then she has a strategy σ with $\mathrm{Cst}(\sigma) \leq n$, i.e., an optimal strategy has cost at most n.*

4 The Complexity of Solving Parity Games with Costs Optimally

In this section we study the complexity of determining the cost of an optimal strategy for a parity game with costs. Recall that solving such games is in UP ∩ CO-UP (and therefore unlikely to be NP-complete or CO-NP-complete) while solving the special case of finitary parity games is in PTIME. Our main result of this section shows that checking whether a strategy of cost at most b exists is PSPACE-complete, where hardness already holds for finitary parity games. Therefore, this decision problem is harder than just solving the game (unless PSPACE ⊆ UP ∩ CO-UP, respectively PSPACE ⊆ PTIME).

▶ **Theorem 3.** *The following problem is* PSPACE*-complete: "Given a parity game with costs $\mathcal{G}$ and a bound $b \in \mathbb{N}$, does Player* 0 *have a strategy σ for $\mathcal{G}$ with $\mathrm{Cst}(\sigma) \leq b$?"*

The proof of this theorem is split into two lemmas, Lemma 4 showing membership and Lemma 7 showing hardness, which are presented in Section 4.1 and Section 4.2, respectively.

4.1 Playing Parity Games with Costs Optimally is in PSPACE

In this section we establish PSPACE-membership of the previously defined decision problem.

▶ **Lemma 4.** *The following problem is in* PSPACE*: "Given a parity game with costs $\mathcal{G}$ and a bound $b \in \mathbb{N}$, does Player* 0 *have a strategy σ for $\mathcal{G}$ with $\mathrm{Cst}(\sigma) \leq b$?"*

The remainder of this section is dedicated to the proof of Lemma 4. To this end, we fix a parity game with costs $\mathcal{G} = (\mathcal{A}, \text{CostParity}(\Omega, \text{Cst}))$ with $\mathcal{A} = (V, V_0, V_1, E, v_I)$ and a bound b. Let $n = |V|$. First, let us remark that we can assume w.l.o.g. $b \leq n$: If $b > n$, then, due to Corollary 2, we just have to check whether Player 0 wins $\mathcal{G}$. This is possible in PSPACE due to Theorem 1.3.

To obtain a polynomial space algorithm, we first turn the quantitative game $\mathcal{G}$ into a qualitative parity game $\mathcal{G}'$ in which the cost of open requests is explicitly tracked up to the bound b . To this end, we use functions r mapping odd colors to $\{\bot\} \cup [b+1]$, where $\bot$ encodes that no open request of this color is pending. Additionally, whenever the bound b is exceeded for some request, all open requests are reset and a so-called overflow counter is increased, up to value n. This accounts for a bounded number of unanswered requests, which are allowed by the parity condition with costs. Intuitively, Player 1 wins $\mathcal{G}'$ if he either exceeds the upper bound b at least n times, or if he enforces an infinite play of finite cost with infinitely many unanswered requests. If he wins by the former condition, this implies that he can also enforce infinitely many excesses of b via a pumping argument. The latter condition accounts for plays in which Player 1 wins without violating the bound b repeatedly, but by violating the classical parity condition. We show that Player 0 has a strategy σ in $\mathcal{G}$ with $\text{Cst}(\sigma) \leq b$ if and only if Player 0 wins $\mathcal{G}'$ from its initial vertex $v_I' = (v_I, r_0, 0)$. Here, r_0 encodes the open requests of the play prefix v_I.

The resulting game $\mathcal{G}'$ is of exponential size in the number of odd colors and can therefore in general not be solved in polynomial space in n. Thus, in a second step, we construct a finite-duration variant $\mathcal{G}_f'$ of $\mathcal{G}'$, which is played on the same arena as $\mathcal{G}'$, but each play is stopped after a polynomial number of moves. We show that Player 0 wins $\mathcal{G}'$ if and only if she wins $\mathcal{G}_f'$.

To conclude, we show how to simulate $\mathcal{G}_f'$ on the fly on an alternating Turing machine in polynomial time in n, which yields a polynomial space algorithm by removing the alternation [6].

We begin by defining $\mathcal{G}'$. Let $R = (\{\bot\} \cup [b+1])^D$ be the set of request functions, where D is the set of odd colors occurring in $\mathcal{G}$. Here, $r(c) = \bot$ denotes that there is no open request for the color c, while $r(c) \neq \bot$ encodes that the oldest open request of c has incurred cost $r(c)$. Using these functions, we define the memory structure $\mathcal{M} = (R \times [n+1], m_I, \text{Upd})$, where the second component $[n+1] = \{0, \ldots, n\}$ implements the overflow counter. It suffices to bound this counter by n, since, if Player 1 can enforce n overflows in $\mathcal{G}$, then, by a pumping argument, he can also enforce infinitely many.

The initial memory state m_I is the pair $(r_I, 0)$, where r_I is the function mapping all odd colors to $\bot$, if $\Omega(v_I)$ is even. If $\Omega(v_I)$ is odd, however, r_I maps $\Omega(v)$ to 0, and all other odd colors to $\bot$. The update function $\text{Upd}(m, e)$ is defined such that traversing an edge $e = (v, v')$ updates $m = (r, o)$ to $m' = (r', o')$ by performing the following steps in order:

- If e is an increment-edge, then for each c with $r(c) \neq \bot$, $r(c)$ is increased by one.
- If there exists a color c such that $r(c) > b$, then reset all open requests to $\bot$ and set o to the minimum of $o+1$ and n.
- If $\Omega(v')$ is even, reset all requests for colors c' with $c' \leq \Omega(v')$ to $\bot$.
- If $\Omega(v')$ is odd, then set $r'(\Omega(v'))$ to the maximum of $r(\Omega(v'))$ and 0, with $\max\{\bot, 0\} = 0$.

The resulting function r' is an element of R and the resulting o' is at most n.

The evolution of the memory states on a play prefix is depicted in Figure 2. The prefix and the sequence of memory states for $b = 2$ are shown in the upper and lower row, respectively. The request functions r are given in vector notation, where the upper and the lower entry denote $r(1)$ and $r(3)$, respectively.

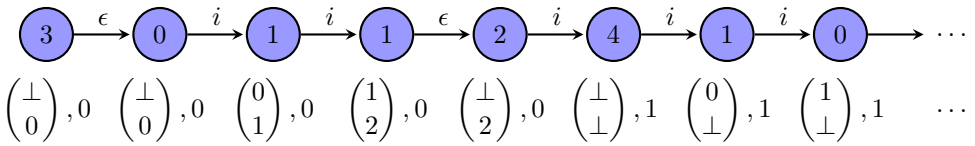


Figure 2 Example of the evolution of the request-functions during a play for the bound $b = 2$.

We define the parity game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \text{Parity}(\Omega'))$, with $\Omega'(v, r, o) = \Omega(v)$ for $o < n$ and $\Omega'(v, r, n) = 1$. Note that every play that encounters a vertex of the form (v, r, n) at some point is winning for Player 1. Although $\mathcal{G}'$ has no cost function, we say that an edge $((v, m), (v', m'))$ of $\mathcal{G}'$ is an increment-edge, if (v, v') is an increment-edge in $\mathcal{G}$, otherwise it is an ϵ-edge.

It suffices to solve $\mathcal{G}'$ to determine whether Player 0 can bound the cost in $\mathcal{G}$ by b.

▶ **Proposition 5.** *There exists a strategy σ for Player 0 in $\mathcal{G}$ with $\text{Cst}(\sigma) \leq b$ if and only if Player 0 wins $\mathcal{G}'$.*

The parity game $\mathcal{G}'$ is of exponential size, since the number of possible request functions is exponential. Hence, explicitly constructing and solving $\mathcal{G}'$ using standard methods does not yield a polynomial space algorithm. Rather, we now show that it suffices to consider a finite-duration variant $\mathcal{G}'_f$ of $\mathcal{G}'$, in which each play ends after a polynomial number of steps. The winner of $\mathcal{G}'_f$ can then be determined by simulating $\mathcal{G}'_f$ on the fly on an alternating polynomial time Turing machine.

We say that a play prefix $(v_0, r_0, o_0) \cdots (v_j, r_j, o_j)$ of $\mathcal{G}'$ is *settled* (for Player 1), if $o_j = n$ or if its projection $v_0 \cdots v_j$ contains a cycle of cost zero whose maximal color is odd. Note that a cycle with odd maximal color that contains an increment-edge does not settle the play, as its existence does not imply the existence of a cycle in $\mathcal{G}'$. Fix $\ell = 3(n+1)^5$. We construct the finite duration variant $\mathcal{G}'_f = (\mathcal{A} \times \mathcal{M}, \text{Win}_\ell)$ of $\mathcal{G}'$, where a play $\rho = (v_0, r_0, o_0)(v_1, r_1, o_1)(v_2, r_2, o_2) \cdots$ is winning for Player 0 if and only if the prefix $(v_0, r_0, o_0) \cdots (v_\ell, r_\ell, o_\ell)$ of length $\ell + 1$ is not settled. Note that $\mathcal{G}'_f$ is indeed a game of finite duration, as the winner is certain after ℓ moves. Hence, $\mathcal{G}'_f$ is determined [26].

▶ **Proposition 6.** *Player 0 wins $\mathcal{G}'$ if and only if she wins $\mathcal{G}'_f$.*

Combining Propositions 5 and 6 implies that Player 0 has a strategy σ for $\mathcal{G}$ with $\text{Cst}(\sigma) \leq b$ if and only if she wins $\mathcal{G}'_f$. Thus it remains to show that we can simulate $\mathcal{G}'_f$ on an alternating Turing machine in polynomial time.

We show how to simulate a play of the finite-duration game $\mathcal{G}'_f$ on an alternating polynomial time Turing machine using the game semantics of such machines, i.e., two players construct a single path of a run of the machine. The existential player takes the role of Player 0, the universal one the role of Player 1. The Turing machine keeps track of the current vertex of the simulated play of $\mathcal{G}'_f$, whether a settling cycle has been encountered, and of the number of moves already simulated. Once ℓ moves have been simulated, the machine terminates and accepts if and only if the play constructed during the run is not settled. To check for zero cycles with odd maximal color, the machine uses the latest-appearance data structure (see, e.g., [16]), which can be updated in linear time and is reset every time an increment-edge is traversed. Note that this algorithm involves neither the explicit construction of $\mathcal{G}'$ nor that of $\mathcal{G}'_f$.

Thus, the Turing machine accepts $\mathcal{G}$ and b if and only if Player 0 wins $\mathcal{G}'_f$. Hence, APTIME = PSPACE [6] completes the proof.

4.2 Playing Parity Games with Costs Optimally is PSPACE-hard

Next, we turn our attention to proving a matching lower bound, which already holds for finitary parity games, i.e., parity games with costs in which every edge is an increment-edge. The result is proven by a reduction from the canonical PSPACE-hard problem QBF: given a quantified boolean formula $\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi$ with $Q_i \in \{\exists, \forall\}$ and where ψ is a boolean formula over the variables $x_1, x_2, \dots, x_n$, determine whether φ evaluates to true. We assume w.l.o.g. that ψ is in conjunctive normal form such that every conjunct has exactly three literals, i.e., $\psi = \bigwedge_{j=1}^{m} (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$, where every $\ell_{j,k}$ is either x_i or $\overline{x_i}$ for some i. We call each $\ell_{j,k}$ for $k \in \{1,2,3\}$ a literal and each conjunct of three literals a clause. Furthermore, we assume w.l.o.g. that the quantifiers Q_i are alternating with $Q_1 = Q_n = \exists$.

▶ **Lemma 7.** *The following problem is* PSPACE*-hard: "Given a finitary parity game $\mathcal{G}$ and a bound $b \in \mathbb{N}$, does Player 0 have a strategy σ for $\mathcal{G}$ with $\mathrm{Cst}(\sigma) \leq b$?"*

Proof. Let $\varphi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi$ be a quantified boolean formula with $\psi = \bigwedge_{j=1}^{m} C_j$ and $C_j = (\ell_{j,1} \vee \ell_{j,2} \vee \ell_{j,3})$, where every $\ell_{j,k}$ is either x_i or $\overline{x_i}$ for some i. We construct a finitary parity game $\mathcal{G}_\varphi$ such that Player 0 has a strategy σ for $\mathcal{G}_\varphi$ with $\mathrm{Cst}_{\mathcal{G}_\varphi}(\sigma) = 3n + 5$ if and only if the formula φ evaluates to true. The arena consists of three parts: In the first part, which begins with the initial vertex v, Player 0 and Player 1 determine an assignment for the variables x_1 through x_n, where Player 0 and Player 1 pick values for the existentially and universally quantified variables, respectively. Each choice of a truth value by either player incurs a request. In the second part, Player 1 first picks a clause, after which Player 0 picks a literal from that clause. In the last part, the play then proceeds without any choice by the players and checks whether or not that literal was set to true in the first part of the arena. If it was set to true, then its corresponding request is answered with cost $3n + 5$. Otherwise, its corresponding request is answered with cost $3n + 6$. Furthermore, all other potentially open requests are answered with cost at most $3n + 5$ and the play returns to the initial vertex v. Thus, all these gadgets are traversed infinitely often and the traversals are independent of each other.

If φ evaluates to true, then Player 0 can always enforce a play in which all requests are answered with cost at most $3n + 5$. Hence, there exists a strategy σ for Player 0 with $\mathrm{Cst}_{\mathcal{G}_\varphi}(\sigma) \leq 3n + 5$. If φ evaluates to false, however, then Player 1 can enforce requests that remain unanswered for at least $3n + 6$ steps. Thus, there exists no strategy σ for Player 0 with $\mathrm{Cst}_{\mathcal{G}_\varphi}(\sigma) \leq 3n + 5$. We begin by constructing the arena $\mathcal{A}$ together with its coloring Ω.

The left-hand side of Figure 3 shows the gadgets that assign a truth-value to variable x_j. The vertex a_j belongs to Player 0 if x_j is existentially quantified, and to Player 1 if x_j is universally quantified. The dashed edges indicate the connections to the pre- and succeeding gadget, respectively. We construct the first part of $\mathcal{A}$ out of n copies of this gadget. Moreover, the vertex a_1 has an incoming edge from the end of $\mathcal{A}$, in order to allow for infinite plays, and is the initial vertex v of the arena. In the remainder of this proof, let $c_{x_j} = 4j + 3$ and $c_{\overline{x_j}} = 4j + 1$ be the colors associated with assigning true or false to x_j, respectively.

The second part of the arena starts with a vertex ψ of Player 1, from which he picks a clause by moving to a vertex C_j of Player 0. Each vertex C_j is connected to three gadgets, one for each of the three literals contained in C_j. We show this construction in the right-hand side of Figure 3. Note that the distance between a vertex of color c_{x_j} and the vertex ψ is $3(n - j) + 1$, whereas the distance between a vertex of color $c_{\overline{x_j}}$ and the vertex ψ is $3(n - j) + 2$.

The last part of the arena consists of one gadget for each literal x_1, $\overline{x_1}$ through x_n, $\overline{x_n}$ occurring in φ. These gadgets check whether or not the literal picked in the middle part was

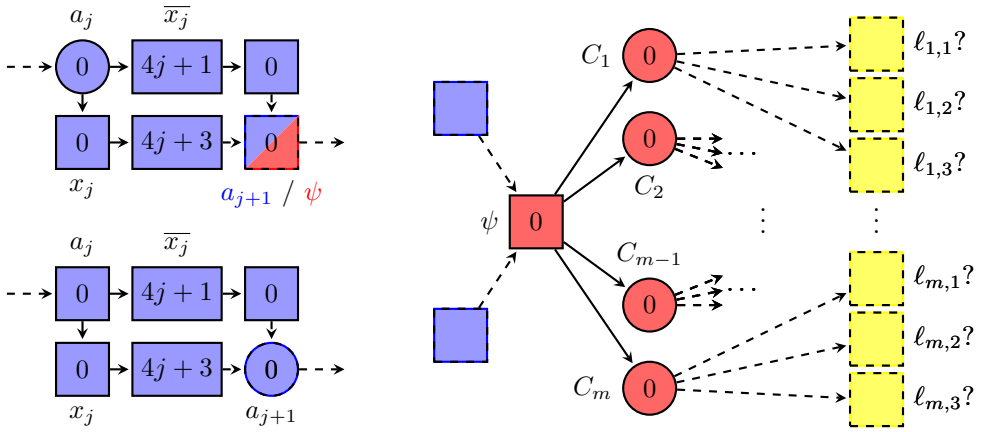


Figure 3 The gadget for existentially and universally quantified variables (left, from top to bottom), and the middle part of the arena (right).

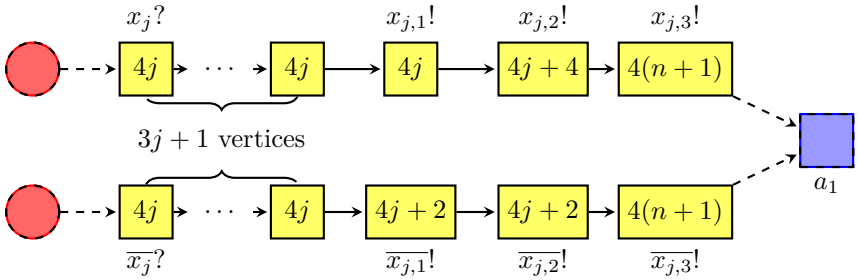


Figure 4 Gadgets checking the assignment of true (top) or false (bottom) to x_j.

actually set to true in the first part of the arena. We show them in Figure 4. Neither player has any control over the play in these gadgets.

Instead, it is determined by their structure. The play proceeds by first answering requests for all colors smaller than c_{x_j} and $c_{\overline{x_j}}$. It then either grants the request for color c_{x_j} after $3j+2$ steps, or the request for color $c_{\overline{x_j}}$ after $3j+1$ steps, both counted from the vertices $x_j?$ and $\overline{x_j}?$, respectively. Since traversing the middle part of the arena incurs a constant cost of 2, a request for color c_{x_j} has incurred a cost of $3(n-j)+3$ at $x_j?$ and $\overline{x_j}?$, while a request for color $c_{\overline{x_j}}$ has incurred a cost of $3(n-j)+4$ at these vertices. Hence, the total cost incurred by the request for color c_{x_j} is $(3(n-j)+3)+(3j+2)$ in the gadget corresponding to x_j, and $(3(n-j)+3)+(3j+3)$ in the gadget corresponding to $\overline{x_j}$. The dual reasoning holds true for requests for color $c_{\overline{x_j}}$. Hence, the bound of $3n+5$ is only achieved if the request corresponding to the chosen literal was posed in the initial part of the arena.

After traversing this last gadget, all requests are answered and the game resets to the initial state via an edge to a_1.

The size of $\mathcal{A}$ is polynomial in the size of φ: The first part consists of one constant-size gadget for each variable, while the second part is of linear size in the number of clauses in φ. The final part contains a gadget of size $\mathcal{O}(n)$ for each literal occurring in φ. Thus, the size of the arena is in $\mathcal{O}(n^2+m)$. It remains to argue that Player 0 has a strategy σ with $\mathrm{Cst}_{\mathcal{G}_\varphi}(\sigma) = 3n+5$ if and only if φ evaluates to true. For any quantifier-free boolean formula ψ that contains variables x_1 through x_n and any partial assignment $\alpha\colon \{x_1,\dots,x_n\} \dashrightarrow \{\mathit{true},\mathit{false}\}$ we denote by $\alpha(\psi)$ the formula resulting from replacing the variables in α's domain with their respective truth values.

It suffices to argue about finite plays that begin and end in a_1, as all plays start in a_1, visit a_1 infinitely often, and all requests are answered before moving back to a_1. Hence, for the remainder of this proof, we only consider a finite play infix ρ starting and ending in a_1.

First assume that φ evaluates to true. We construct a strategy σ for Player 0 with the properties described above. Pick j as some index such that x_j is existentially quantified and consider the play prefix ρ' of ρ up to, but not including a_j. We associate ρ' with an assignment $\alpha_{j-1}\colon \{x_1,\dots,x_{j-1}\} \to \{\mathit{true},\mathit{false}\}$, where $\alpha_{j-1}(x_k) = \mathit{true}$ if there is an infix $a_k x_k$ in ρ', and $\alpha_{j-1}(x_k) = \mathit{false}$ if there is an infix $a_k\overline{x_k}$ in ρ'. Due to the structure of the arena exactly one of these cases holds true, hence α_{j-1} is well-defined.

For $j = 1$, $\exists x_j \cdots Q_n x_n \alpha_{j-1}(\psi)$ evaluates to true by assumption. Let $t \in \{\mathit{true},\mathit{false}\}$ such that $\forall x_{j+1} \cdots Q_n x_n(\alpha_{j-1}[x_j \mapsto t])(\psi)$ evaluates to true as well, where $\alpha_{j-1}[x_j \mapsto t]$ denotes the mapping α_{j-1} augmented by the mapping $x_j \mapsto t$. Moreover, we define $\sigma(wa_j) = x_j$ if $t = \mathit{true}$, and $\sigma(wa_j) = \overline{x_j}$ otherwise. We proceed inductively, constructing $\sigma(wa_j)$ for all existentially quantified variables x_j according to the boolean values that satisfy the formulas $\exists x_j Q_{j+1} x_{j+1} \dots Q_n x_n \alpha_{j-1}(\psi)$, until we reach the vertex ψ.

At this point, the analysis of the play prefix so far yields an assignment α_n, which is a total function mapping each variable of x_1 through x_n to either *true* or *false*, such that $\alpha_n(\psi)$ evaluates to true. We define $\alpha = \alpha_n$.

At vertex ψ there exist n open requests. As previously argued, if $\alpha(x_j) = \mathit{true}$, then there is an open request for c_{x_j} with cost $3(n-j)+1$. Otherwise, there is an open request for $c_{\overline{x_j}}$ with cost $3(n-j)+2$. At vertex ψ, Player 1 picks a clause C_j by moving to its vertex. Since $\alpha(\psi) \equiv \mathit{true}$, there must exist a $k \in \{1,2,3\}$ with $\alpha(\ell_{j,k}) = \mathit{true}$. We pick $\sigma(wC_j) = \ell_{j,k}?$.

If $\ell_{j,k} = x_l$, then $\alpha(x_l) = \mathit{true}$ and hence, there is an open request for c_{x_j}. As argued previously, this request is then answered with cost $3n+5$, since we picked the gadget corresponding to x_j. Similarly, if $\ell_{j,k} = \overline{x_l}$, then $\alpha(x_l) = \mathit{false}$ and thus there is an open request for $x_{\overline{x_j}}$, which is answered with cost $3n+5$ as well. All other open requests are answered with cost at most $3n+5$, as argued previously.

After this traversal of the final gadget, all requests are answered, and the play automatically moves to vertex a_1 to begin anew. The same reasoning then applies ad infinitum. Thus, Player 0 is able to answer all requests with a cost of at most $3n+5$.

Now assume that φ evaluates to false. Then, irrespective of the choices made by Player 0 when constructing α in the first part of the arena, Player 1 can construct an α such that $\alpha(\psi) \equiv \mathit{false}$ and then pick a clause C_j with $\alpha(C_j) \equiv \mathit{false}$. Hence, Player 0 has to pick some $\ell_{j,k}$ with $\alpha(\ell_{j,k}) = \mathit{false}$. If $\ell_{j,k} = x_l$, then there is an open request for $c_{\overline{x_j}}$ at $x_{l,1}!$, which is answered with cost $3n+6$. Similarly, if $\ell_{j,k} = \overline{x_l}$, then $\alpha(x_l) = \mathit{true}$, hence there is an open request for c_{x_j}, which is also answered with cost $3n+6$. Thus, in each round Player 1 can open a request that is answered with cost at least $3n+6$, i.e., Player 0 has no strategy with cost $3n+5$. ◀

5 Memory Requirements of Optimal Strategies in Parity Games with Costs

Next, we study the memory requirements of optimal strategies in parity games with costs. Recall that Player 0 always has a positional winning strategy if she wins the game. In contrast, our main result of this section shows that the memory requirements of optimal strategies are exponential, i.e., playing optimally comes at a price in terms of memory, too. Our lower bound is obtained by a generalization of a construction of Chatterjee and Fijalkow [8] which yielded a linear lower bound.

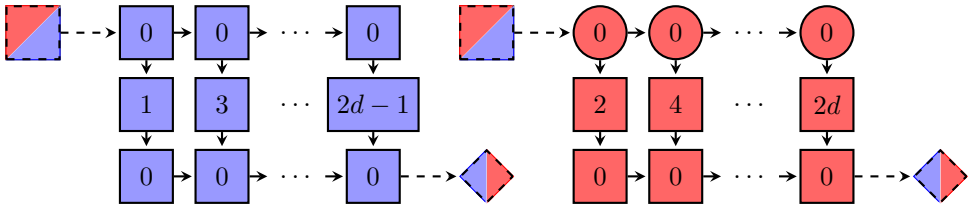


Figure 5 The gadgets for Player 1 (left) and Player 0 (right).

First, however, let us state a corollary of the construction of the parity game $\mathcal{G}'$ in the proof of Lemma 4, which gives an exponential upper bound on the necessary memory states. Recall that the memory structure used in that proof has one counter with a range of size $b+2$ for each odd color. Furthermore, it has an additional counter that is bounded by n, which counts the number of times the bound b is exceeded.

▶ **Corollary 8.** *Let $\mathcal{G}$ be a parity game with costs. If Player 0 has a strategy σ for $\mathcal{G}$ with $\mathrm{Cst}_{\mathcal{G}}(\sigma) = b$, then she also has a strategy σ' with $\mathrm{Cst}_{\mathcal{G}}(\sigma') \leq b$ and $|\sigma'| = (b+2)^d \cdot (n+1)$, where n is the number of vertices and d the number of odd colors in $\mathcal{G}$.*

Using similar techniques to [15], it is possible to remove the overflow counter, since it is the goal of Player 0 to avoid excesses of the bound b. Hence, she can play assuming the largest value for this counter that still allows her to win. Again, our matching lower bound already holds for finitary parity games, i.e., parity games with costs in which all edges are increment-edges.

▶ **Theorem 9.** *For every $d > 1$, there exists a finitary parity game $\mathcal{G}_d$ such that*

- *$\mathcal{G}_d$ has d odd colors and $|\mathcal{G}_d| \in \mathcal{O}(d^2)$, and*
- *every optimal strategy for Player 0 in $\mathcal{G}_d$ has at least size $2^d - 2$.*

Proof. Let $d > 1$. We construct a finitary parity game $\mathcal{G}_d$ that has the stated properties. To this end, we first construct an optimal strategy for Player 0 and argue that every such strategy has cost $d^2 + 2d$, followed by the proof that every optimal strategy has at least size $2^d - 2$.

The game $\mathcal{G}_d$ is played in rounds. In each round, which starts at the initial vertex of $\mathcal{G}_d$, Player 1 poses d requests for odd colors in the range 1 through $2d-1$. Subsequently, Player 0 gives d answers using colors in the range 2 through $2d$. After each round, the play returns to the initial vertex in order to allow for infinite plays.

The arena $\mathcal{A}$ consists of gadgets that allow exactly one request or response to be made. Moreover, each path through a gadget has the same length. However, low-priority requests and responses must be made earlier in the traversal of such a gadget than high-priority ones. We show both gadgets in Figure 5. The dashed lines show the connection to the pre- and succeeding gadget and the connection between the final and the initial gadget. As the owner of the succeeding vertex depends on the succeeding gadget's owner, we draw it as a diamond.

The arena $\mathcal{A}$ consists of d repetitions of the gadget for Player 1, followed by d repetitions of the gadget for Player 0. The initial vertex v of the arena is the top-left node of the first gadget for Player 1. Moreover, the final gadget of Player 0 has a single back-edge to the initial vertex. Clearly, $\mathcal{A}$ satisfies the first statement of the theorem.

Similarly to the proof of Lemma 7, it suffices to consider finite plays infixes. Even though the requests are not necessarily all answered after each round, we argue that Player 0 can always do so while playing optimally.

We now construct an optimal strategy from v for Player 0. In order to play optimally, Player 0 needs to track the requests made by Player 1 in the first part of each round. Instead of tracking each request precisely, however, it suffices to only store those requests that are of higher priority than all previous ones in the current round. Moreover, by defaulting to visiting the vertex of color $2d$ after having answered all requests, it suffices to store at most $d-1$ requests. If there is a repetition in the requests made, then the final request will be answered by an answer previous to the final one. If there is no repetition, the final request is for $2d-1$ and will be answered by the default answer of $2d$. We define the set of strictly increasing odd sequences $\mathit{IncSeq}_d = \{(c_1, \ldots, c_k) \mid 1 \leq c_1 < \cdots < c_k \leq 2d-1, \text{ all } c_i \text{ are odd}\}$ and use the set of memory states $M_d = \mathit{IncSeq}_d \setminus \{(), (1, 3, \ldots, 2d-1)\}$. Note that $|M_d| = 2^d - 2$.

Each round starts with Player 0 observing the requests of Player 1. While the first request of Player 1 replaces the entry of the initial memory state (1), subsequent requests are only appended if they are larger than the last element of the current memory state. During her turn, Player 0 then pops the first element of the sequence, call it c, in each of her gadgets and answers that request by moving to $c+1$. After popping the final request in her memory state, Player 0 pushes a request for $2d-1$ and answers with $2d$ for the remainder of her turn.

Now consider a play in which Player 0 plays according to this strategy and consider the request for color c made by Player 1 in his j-th gadget. If Player 1 has posed strictly increasing requests up to j, then Player 0 answers the j-th request from Player 1 in her j-th gadget. The cost of this request then consists of three components. First, the play has to leave Player 1's j-th gadget, incurring a cost of $(2d-1-c)/2+2$. Then, the play passes through $d-1$ gadgets, each incurring a cost of $d+2$. Finally, moving to $c+1$ in Player 0's gadget incurs a cost of $(c-1)/2+1$ Hence, answering Player 1's request incurs a cost of d^2+2d. If Player 1 has, however, not posed requests in strictly increasing order up to j, then the request will be answered in some gadget $j' < j$ of Player 0. Hence, Player 0 answers the request with cost at most $(d-1)(d+2) < d^2+2d$.

This cost is indeed optimal. Consider the play in which Player 1 always requests $2d-1$. Even if Player 0 answers this request in her first gadget, it still incurs a cost of d^2+2d.

It remains to show that no optimal strategy of size less than $|M_d|$ exists. We associate with each $s \in M_d$ the partial play $\mathit{req}(s)$, which starts in the initial vertex, where Player 1 requests the colors occurring in s in order. Subsequently, he requests the final color of s until the end of his turn. Hence, for $s \neq s' \in M_d$, we have $\mathit{req}(s) \neq \mathit{req}(s')$.

Pick a strategy σ for Player 0 that is implemented by $\mathcal{M} = (M, m_I, \mathrm{Upd})$ with $|M| < |M_d|$. Let $m \in M$. Due to the pigeon-hole principle, there exist $s \neq s' \in M_d$, such that $\mathrm{Upd}^+(m, \mathit{req}(s)) = \mathrm{Upd}^+(m, \mathit{req}(s'))$. Hence, Player 0 answers both sequences of requests in the same way. Since $\mathit{req}(s) \neq \mathit{req}(s')$, there exists a gadget of Player 1 in which the requests posed during $\mathit{req}(s)$ and $\mathit{req}(s')$ differ. Pick j as the minimal index of such gadgets and assume that in his j-th gadget, Player 1 requests color c during $\mathit{req}(s)$, and color c' during $\mathit{req}(s')$, where, w.l.o.g., $c < c'$. If Player 0 has already answered either the request for c or the request for c' before her j-th gadget, then some earlier request is not answered optimally. Thus, assume that neither request has been answered upon entering Player 0's j-th gadget. If she visits some color $c'' < c'$ in this gadget, she will only answer c' in some later gadget, thereby incurring a cost of at least $(d+1)(d+2)$. If she visits some color $c'' > c'$, then she does not answer the request for c optimally, thus incurring a cost of at least $d^2+2d+c'-c$. Hence, one of the sequences of requests s or s' leads to Player 0 answering at least one request non-optimally. As there exist such sequences s and s' for each $m \in M$, Player 1 can force such a costly request in each round. Thus, $\mathrm{Cst}(\sigma) > d^2+2d$, i.e., σ is not optimal. ◀

6 Tradeoffs Between Time and Memory

In the previous section, we have shown that an optimal strategy for Player 0 in a parity game with costs requires exponential memory in general. In contrast, minimal winning strategies for Player 0 in parity games with costs are known to be positional [15]. Here we show that, in general, there exists a gradual tradeoff between the size and the cost of a strategy.

▶ **Theorem 10.** *Fix some $d > 1$ and let the game $\mathcal{G}_d$ be as defined in the proof of Theorem 9. For every i with $1 \leq i \leq d$ there exists a strategy σ_i for Player 0 in $\mathcal{G}_d$ such that*

- $d^2 + 3d - 1 = \mathrm{Cst}_{\mathcal{G}_d}(\sigma_1) > \mathrm{Cst}_{\mathcal{G}_d}(\sigma_2) > \cdots > \mathrm{Cst}_{\mathcal{G}_d}(\sigma_d) = d^2 + 2d$, *and*
- $1 = |\sigma_1| < |\sigma_2| < \cdots < |\sigma_d| = 2^d - 2$.

Also, for every strategy σ' for Player 0 in $\mathcal{G}_d$ with $\mathrm{Cst}_{\mathcal{G}_d}(\sigma') \leq \mathrm{Cst}_{\mathcal{G}_d}(\sigma_i)$ we have $|\sigma'| \geq |\sigma_i|$.

Proof. Recall that we defined the set of strictly increasing odd sequences IncSeq_d in Theorem 9 and showed that a memory structure using $\mathit{IncSeq}_d \setminus \{(), (1, 3, \ldots, 2d-1)\}$ as memory states implements an optimal strategy with cost $d^2 + 2d$. Intuitively, such a strategy stores up to $d - 1$ requests made by Player 1 in his part of each round. The idea behind the construction of the strategies σ_i is to restrict the memory of Player 0 such that she can only store up to i requests. In the extremal cases of $i = 1$ and $i = d$ this implements a positional strategy and the strategy from the proof of Theorem 9, respectively.

We implement σ_i by again using strictly increasing odd sequences, where we restrict the maximal length, but not the maximal value of entries in the memory states for Player 0. In strategy σ_i, Player 0 stores at most $i - 1$ requests, using sequences of length at most $i - 1$.

To this end, we define the length-restricted set of strictly increasing odd sequences $\mathit{IncSeq}_d^i = \{s \in \mathit{IncSeq}_d \mid |s| < i\}$, where $|s|$ denotes the number of elements contained in the sequence s. If $i > 1$, then we may remove the empty sequence from the set of memory states, since Player 1 has to pose at least one request in each of his turns.

Otherwise, Player 0's memory consists only of the empty sequence.

Hence, we pick $M_d^i = \mathit{IncSeq}_d^i \setminus \{()\}$ for $i > 1$. For $i = 1$, however, we define $M_d^1 = \mathit{IncSeq}_d^1 = \{()\}$. Note that $M_d^d = M_d$ as defined in the proof of Theorem 9. Clearly, the claim about the size of the σ_j holds true, since $|M_d^1| < |M_d^2|$ and $M_d^i \subsetneq M_d^{i+1}$ for each $d > 1$, $i > 1$. The initial memory state, the update function, and the next-move function for Player 0 are defined similarly to those from the proof of Theorem 9 in order to obtain the memory structure $\mathcal{M}_i$ implementing σ_i. In particular, after answering all requests stored in her memory state, Player 0 defaults to visiting $2d$ repeatedly in order to answer any requests by Player 1 that were not stored in her memory.

It remains to show that each strategy σ_j realizes a cost of $d^2 + 3d - i$ and that each σ_i is minimal for its respective cost. To this end, we fix some i with $1 \leq i \leq d$ for the remainder of this proof. First, we show that Player 1 can enforce a cost of $d^2 + 3d - i$ if Player 0 plays consistently with σ_i. Intuitively, Player 1 fills the memory of Player 0 as quickly as possible, and requests the minimal color that has not yet been requested repeatedly afterwards. Thus, he maximizes the gap between the smallest unstored request and the default answer of $2d$.

More precisely, in each turn Player 1 requests the colors $1, 3, \ldots, 2i-3, 2i-1, 2i-1, \ldots$. Playing consistently with σ_i, Player 0 answers these requests with $2, 4, \ldots, 2i-2, 2d, 2d, \ldots$. In general,the cost of the resulting play is the cost incurred by answering a request for $2i - 1$ in the i-th gadget of Player 1 with $2d$ in the i-th gadget of Player 0. As argued in the proof of Theorem 9, the cost incurred by such a request-response-pair amounts to

$$[(2d - 1 - (2i - 1))/2 + 2] + [(d - 1)(d + 2)] + [(2d - 2)/2 + 1] = d^2 + 3d - i.$$

As the game restarts after Player 0's turn, Player 1 can enforce this cost infinitely often. Hence, $\mathrm{Cst}_{\mathcal{G}_d}(\sigma_i) \geq d^2 + 3d - i$.

This sequence of requests is indeed optimal for Player 1, i.e., he cannot enforce a higher cost. Assume that Player 1 does not pose requests as specified above, but poses the requests $c_1, \ldots, c_d$. Then either there exist some j and j' with $j < j'$, such that $c_j \geq c_{j'}$, or there exists a j with $2i - 1 < c_j \leq 2d - 1$. In the former case, after encountering the first such j', Player 0 can answer all remaining requests with costs at most $(d-1)(d+2)$, as she can ignore the request for $c_{j'}$. In the latter case, Player 0 answers that request with cost at most $d^2 + 2d + (2d - 1 - c_j)/2 \leq d^2 + 3d - i$, independent of whether she was able to store it. Hence, there exists no play ρ consistent with σ_i and $\mathrm{Cst}(\rho) > d^2 + 3d - i$.

As the final part of the proof, we observe that there exists no strategy σ' with $|\sigma'| < |\sigma_i|$ and $\mathrm{Cst}_{\mathcal{G}_d}(\sigma') \leq \mathrm{Cst}_{\mathcal{G}_d}(\sigma_i)$. The argument is nearly identical to the argument of minimality of the strategy constructed in the proof of Theorem 9 and can in fact be obtained by replacing all occurrences of $2^d - 2$ and $d^2 + 2d$ by $|\sigma_i|$ and $d^2 + 3d - i$, respectively, and by having Player 1 request $2i - 1$ instead of the final entry of s after requesting all colors in s in that proof. Hence, the strategies σ_i are minimal for their respective cost. ◀

7 Conclusion

In this work we have shown that playing parity games with costs optimally is harder than just winning them, both in terms of computational complexity as well as in terms of memory requirements of strategies. We proved checking an upper bound on the value of an optimal strategy to be complete for polynomial space. Moreover, we have shown that optimal strategies in general require exponential memory, but also that exponential memory is always sufficient to implement optimal strategies. Finally, we have shown that, in general, there exists a gradual tradeoff between the size and the cost of strategies.

All these results also hold true for the case of bounded parity games and bounded parity games with costs [9, 15]. While the parity condition with costs only restricts the cost-of-response in the limit, the bounded parity condition prohibits any unanswered request with cost ∞ (but still allows finitely many unanswered requests with finite cost).

There are at least two directions in which these results can be extended, namely towards Streett conditions (finitary or with costs) [9, 15] and towards a binary weights.

In current work, we investigate the former extension. Our lower bounds carry over trivially, as every parity condition is a Streett condition of polynomial size. On contrast, the upper bounds are more involved, since solving the boundedness question for finitary Streett games is already ExpTime-complete and exponential memory is necessary for Player 0 (see [15] for a discussion).

Throughout this work, we have only considered unary weights, i.e., cost functions that assign the abstract costs ϵ and i. Allowing arbitrary non-negative costs would constitute an extension of the model considered here, i.e., the PSpace-hardness result as well as the necessity of exponential memory remain true without any modifications. Again, the upper bounds are non-trivial, as the upper bound on the cost of an optimal strategy is now exponential in the size of the game and its largest weight. Thus, e.g., there is a blowup in the size of $\mathcal{G}'$ as constructed in the proof of Theorem 4 and in the play length of its finite-duration variant $\mathcal{G}'_f$. We are currently investigating whether these can be avoided.

References

1 Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for "model measuring". *ACM Trans. Comput. Log.*, 2(3):388–407, 2001.

2 Benjamin Aminof and Sasha Rubin. First cycle games. In Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi, editors, *SR 2014*, volume 146 of *EPTCS*, pages 83–90, 2014.

3 Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *FORMATS 2008*, volume 5215 of *LNCS*, pages 33–47. Springer, 2008.

4 Véronique Bruyère, Quentin Hautem, and Mickael Randour. Window parity games: an alternative approach toward parity games with time bounds. *arXiv*, 1606.01831, 2016.

5 Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *EMSOFT 2003*, volume 2855 of *LNCS*, pages 117–133. Springer, 2003.

6 Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

7 Krishnendu Chatterjee, Luca de Alfaro, and Thomas A. Henzinger. Trading memory for randomness. In Gethin Norman and William Sanders, editors, *QEST 2004*, pages 206–217. IEEE, 2004.

8 Krishnendu Chatterjee and Nathanaël Fijalkow. Infinite-state games with finitary conditions. *arXiv*, 1301.2661, 2013.

9 Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in ω-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009. doi:10.1145/1614431.1614432.

10 Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Trading infinite memory for uniform randomness in timed games. In Magnus Egerstedt and Bud Mishra, editors, *HSCC 2008*, volume 4981 of *LNCS*, pages 87–100. Springer, 2008.

11 Andrzej Ehrenfeucht and Jan Mycielski. Positional strategies for mean payoff games. *Int. J. Game Theory*, 8:109–113, 1979.

12 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS 1991*, pages 368–377. IEEE, 1991.

13 Peter Faymonville and Martin Zimmermann. Parametric linear dynamic logic. In Adriano Peron and Carla Piazza, editors, *GandALF 2014*, volume 161 of *EPTCS*, pages 60–73, 2014.

14 Nathanaël Fijalkow, Florian Horn, Denis Kuperberg, and Michal Skrzypczak. Trading bounds for memory in games with counters. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015, Part II*, volume 9135 of *LNCS*, pages 197–208. Springer, 2015.

15 Nathanaël Fijalkow and Martin Zimmermann. Parity and Streett Games with Costs. *LMCS*, 10(2), 2014.

16 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

17 Florian Horn, Wolfgang Thomas, Nico Wallmeier, and Martin Zimmermann. Optimal strategy synthesis for request-response games. *RAIRO – Theor. Inf. and Applic.*, 49(3):179–203, 2015.

18 Marcin Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998. doi:10.1016/S0020-0190(98)00150-1.

19 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Form. Met. in Sys. Des.*, 34(2):83–103, 2009. doi:10.1007/s10703-009-0067-z.

20 Martin Lang. Resource reachability games on pushdown graphs. In Anca Muscholl, editor, *FOSSACS 14*, volume 8412 of *LNCS*, pages 195–209. Springer, 2014.

21 Fabio Mogavero, Aniello Murano, and Loredana Sorrentino. On promptness in parity games. *Fundam. Inform.*, 139(3):277–305, 2015.

22 Andrzej Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.

23 Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP 1989*, volume 372 of *LNCS*, pages 652–671. Springer, 1989. doi:10.1007/BFb0035790.

24 Mickael Randour. *Synthesis in Multi-Criteria Quantitative Games*. PhD thesis, University of Mons, 2014.

25 Alexander Weinert and Martin Zimmermann. Easy to win, hard to master: Optimal strategies in parity games with costs. *arXiv*, 1604.05543, 2016.

26 Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proc. Fifth Congress of Mathematicians, Vol. 2*, pages 501–504. Cambridge Press, 1913.

27 Martin Zimmermann. Time-optimal winning strategies for poset games. In Sebastian Maneth, editor, *CIAA 2009*, volume 5642 of *LNCS*, pages 217–226. Springer, 2009.

28 Martin Zimmermann. Optimal Bounds in Parametric LTL Games. *Theoret. Comput. Sci.*, 493(0):30–45, 2013.

29 Martin Zimmermann. Parameterized linear temporal logics meet costs: Still not costlier than LTL. In Javier Esparza and Enrico Tronci, editors, *GandALF 2015*, volume 193 of *EPTCS*, pages 144–157, 2015.

30 Uri Zwick and Mike Paterson. The complexity of mean payoff games. In Ding-Zhu Du and Ming Li, editors, *COCOON 1995*, volume 959 of *LNCS*, pages 1–10. Springer, 1995.

A Sequent Calculus for a Modal Logic on Finite Data Trees

David Baelde[1], Simon Lunel[2], and Sylvain Schmitz[3]

1 **LSV, ENS Cachan & CNRS & Inria, Université Paris-Saclay, France**
2 **Inria Rennes, France**
3 **LSV, ENS Cachan & CNRS & Inria, Université Paris-Saclay, France**

Abstract

We investigate the proof theory of a modal fragment of XPath equipped with data (in)equality tests over finite data trees, i.e. over finite unranked trees where nodes are labelled with both a symbol from a finite alphabet and a single data value from an infinite domain. We present a sound and complete sequent calculus for this logic, which yields the optimal PSPACE complexity bound for its validity problem.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Complexity of Proof Procedures, F.4.1 [Mathematical Logic] Modal Logic, H.2.3 [Languages] Query languages

Keywords and phrases XPath, proof systems, modal logic, complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.32

1 Introduction

Arguably the most widespread language for querying XML documents, XPath allows to select and extract elements and values from XML documents. It is embedded in the XSLT and XQuery languages and implemented through libraries in many general-purpose programming languages. The language in its successive revisions has evolved into a full-fledged programming language [22], but its distinguishing feature remains a navigational core (known as **CoreXPath** [5]) supplemented with the ability to perform data joins – this is captured in the fragment dubbed **CoreDataXPath** in [7].

Static analysis of XPath queries, typically inclusion or equivalence checks between queries, can be performed formally through the *validity problem* – or equivalently, for those XPath fragments with negation, through the satisfiability problem. In the data-oblivious case, satisfiability is decidable for **CoreXPath** even in the presence of DTDs [3]. The data-aware version **CoreDataXPath** however turns out to be undecidable [4], which has initiated a quest for decidable fragments and variants [7, 17, 15, 11, 12, 13] – often with prohibitively high complexities. This line of work relies on model-theoretic reasoning, and on the development of ad-hoc models of data automata tailored to capture the fragment at hand.

In this paper, we explore a different avenue, namely the usage of proof systems to analyse XPath queries. Proof systems provide indeed much more concrete *proof search* algorithms for query analysis than the typical 'find non-deterministically a model of size $f(n)$' algorithms that result from the model-theoretic and automata-theoretic approaches. They open the door to a wealth of formal results and practical heuristics and optimisations developed over the years for proof search techniques.

In the case of the data-oblivious **CoreXPath**, there are already several Hilbert-style axiomatisations of fragments [4, 24] and extensions with XPath 2.0 features [25]. By

© David Baelde, Simon Lunel, and Sylvain Schmitz;
licensed under Creative Commons License CC-BY

contrast, in this work we do not focus on the navigational aspects of XPath, but rather on understanding how to handle data tests through proof systems. Furthermore, while Hilbert-style axiomatisations provide purely syntactic rules to check the validity of formulæ, decidability and complexity results are rather derived from Gentzen-style sequent calculi or from tableaux systems, and we choose to work with the former.

More precisely, we present a sound and complete cut-free sequent calculus for a fragment of **CoreDataXpath**. For this first attempt at a proof system for a data-aware logic, we work in a somewhat simplified setting:

- our models are finite *data trees* rather than XML trees: these are ordered, unranked trees where each node carries exactly one datum from some infinite data domain $\mathbb{D}$ in addition to a label from some finite alphabet – whereas XML nodes might carry several data values –, and
- we strip **CoreDataXPath**'s navigational capabilities down to a simple *modal data logic* **DataGL** where the usual '$\Box$' modality is refined into two data-aware modalities: $\Box_=$ relates the current position to strict descendants labelled with the same data value, while $\Box_\neq$ relates it to strict descendants with a different data value (see Section 2).
  Our logic **DataGL** is a fragment of **CoreDataXPath**($\downarrow^+$), where navigation is restricted to the strict descendant axis $\downarrow^+$ (see [12]). As already noted by ten Cate, Fontaine, and Litak [23], the similarly defined data-oblivious **CoreXPath**($\downarrow^+$) corresponds naturally to the *provability logic* **GL** (named after Gödel and Löb). Although **GL** was originally intended to model provability in arithmetic, it is best understood for our purpose as the modal logic of finite trees: its set of axioms is sound and weakly complete for well-founded transitive frames (e.g. [6], Chapter 4, where the logic is called **KL**). By the same token, **DataGL** can be seen as the data-aware extension of **GL**.

Our calculus, defined and shown sound and complete with respect to finite data trees in Section 3, builds upon an existing sequent calculus for **GL** defined by Avron [2]. We found nonetheless that dealing with $\Box_\neq$ modalities brought significant new challenges – both when enforcing well-foundedness and when dealing with the non-transitivity of the associated 'descendant with different data' relation –, which we tackle by introducing so-called *histories* in the calculus.

Among the benefits of our calculus, we exhibit a complete proof search strategy in Section 4, and we show that this strategy works in PSPACE in Section 4.3. This is an improvement over the much more general upper bound shown by Figueira [12, Theorem 6.4] for the EXP-complete **CoreDataXPath**($\downarrow^+$), and matches the PSPACE-hardness of **GL** in the data-oblivious case (e.g. [9, Theorem 7]) – thus in **GL**, data can be added for free! Although there might be simpler ways to prove the PSPACE-completeness of **DataGL**, this shows that proof-theoretic methods do not necessarily come at the expense of algorithmic efficiency.

Due to space constrains, some material is omitted but can be found in the full paper at `https://hal.inria.fr/hal-01191172`.

2 Modal Logic on Finite Data Trees

We introduce in this section **DataGL**, a bimodal logic (see Section 2.2) defined over *finite data trees* (recalled first in Section 2.1). The logic **DataGL** has however a natural, equivalent formulation in terms of finite transitive irreflexive *data Kripke structures*, as shown in Section 2.2.2, allowing to reuse the model-theoretic tool set of modal logic – whereas similar results for **CoreDataXPath** fragments are rather more involved [14].

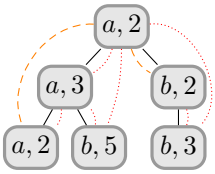


Figure 1 A finite data tree over $\Sigma \stackrel{\text{def}}{=} \{a, b\}$ and $\mathbb{D} \stackrel{\text{def}}{=} \mathbb{N}$. The $R_=$ relation is indicated through dashed orange arcs, and the $R_{\neq}$ relation through dotted red arcs.

2.1 Data Trees

A finite (ordered and unranked) *tree* $\mathfrak{t}$ over an alphabet $\mathbb{A}$ is a partial function from *positions* w in $\mathbb{N}^*$ (i.e. finite sequences of non-negative integers) to $\mathbb{A}$, with a finite non-empty domain $\operatorname{dom}\mathfrak{t}$, which is furthermore

prefix-closed: if $wv \in \operatorname{dom}\mathfrak{t}$ for some $w, v \in \mathbb{N}^*$, then $w \in \operatorname{dom}\mathfrak{t}$, and

predecessor-closed: if $w(i+1) \in \operatorname{dom}\mathfrak{t}$ for some $w \in \mathbb{N}^*$ and $i \in \mathbb{N}$, then $wi \in \operatorname{dom}\mathfrak{t}$.

Call the length $|w|$ the *height* of position w. The maximal such height $h(\mathfrak{t}) \stackrel{\text{def}}{=} \max_{w \in \operatorname{dom}\mathfrak{t}} |w|$ is called the *height* of $\mathfrak{t}$; this is well-defined since $\operatorname{dom}\mathfrak{t}$ is finite. The root of a tree $\mathfrak{t}$ is then denoted by the empty sequence ε, with height 0.

Let Σ be a finite set of tags and $\mathbb{D}$ an infinite countable set of data values. A finite *data tree* is a finite tree over the Cartesian product $\Sigma \times \mathbb{D}$; see Figure 1 for an example. For a position w in $\operatorname{dom}\mathfrak{t}$, we write $\ell(w)$ for its tag in Σ and $d(w)$ for its datum in $\mathbb{D}$; then $\mathfrak{t}(w) = (\ell(w), d(w))$. Given a data tree $\mathfrak{t}$, its *strict descendant* relation $R \stackrel{\text{def}}{=} \{(w, wv) \in \operatorname{dom}\mathfrak{t} \times \operatorname{dom}\mathfrak{t} \mid v \in \mathbb{N}^+\}$ between its positions can be partitioned into $R = R_= \uplus R_{\neq}$ by defining

$$R_= \stackrel{\text{def}}{=} \{(w, w') \in R \mid d(w) = d(w')\}\,, \qquad R_{\neq} \stackrel{\text{def}}{=} \{(w, w') \in R \mid d(w) \neq d(w')\}\,. \tag{1}$$

It is worth noting that $R_=$ is transitive, but $R_{\neq}$ is not – as seen for instance on the leftmost branch of the tree in Figure 1 –; however, $w\,R_{\neq}\,w'\,R_=\,w''$ or $w\,R_=\,w'\,R_{\neq}\,w''$ implies $w\,R_{\neq}\,w''$, a fact we dub *cross transitivity* – see for instance the rightmost branch of the tree in Figure 1.

2.2 Modal Data Logic

Our modal data logic **DataGL** is syntactically a modal logic with two modal operators, namely $\Box_=$ and $\Box_{\neq}$. Given a countable set A of atomic propositions, its set of formulæ is defined by the abstract syntax

$$\varphi ::= \bot \mid p \mid \varphi \supset \varphi \mid \Box_= \varphi \mid \Box_{\neq} \varphi$$

where p ranges over A. The usual Boolean connectives can be defined by $\neg\varphi \stackrel{\text{def}}{=} \varphi \supset \bot$, $\top \stackrel{\text{def}}{=} \neg\bot$, $\varphi \vee \psi \stackrel{\text{def}}{=} (\neg\varphi) \supset \psi$, $\varphi \wedge \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \vee \neg\psi)$, and the diamonds by $\Diamond_=\varphi \stackrel{\text{def}}{=} \neg\Box_=\neg\varphi$ and $\Diamond_{\neq}\varphi \stackrel{\text{def}}{=} \neg\Box_{\neq}\neg\varphi$; finally the usual box and diamond are defined through $\Box\varphi \stackrel{\text{def}}{=} \Box_=\varphi \wedge \Box_{\neq}\varphi$ and $\Diamond\varphi \stackrel{\text{def}}{=} \neg\Box\neg\varphi$. Importantly, $\Diamond_=$ and $\Box_{\neq}$ are *not* dual, nor are $\Diamond_{\neq}$ and $\Box_=$.

(a) A counter-model to (2). **(b)** An infinite counter-model to (3).

Figure 2 Counter-models to formulæ (2) and (3), where $\varphi \stackrel{\text{def}}{=} p$, $\Sigma \stackrel{\text{def}}{=} \{\emptyset, \{p\}\}$, and $\mathbb{D} \stackrel{\text{def}}{=} \mathbb{N}$.

2.2.1 Semantics

Given a finite data tree $\mathfrak{t}$ and a position $w \in \operatorname{dom} \mathfrak{t}$, we inductively define a **DataGL** formula φ to be *satisfied* in $\mathfrak{t}$ at w, denoted $\mathfrak{t}, w \models \varphi$, as usual:

$$
\begin{aligned}
&\mathfrak{t}, w \models \bot && \text{never ,}\\
&\mathfrak{t}, w \models p && \text{iff } p \in \ell(w) \text{ ,}\\
&\mathfrak{t}, w \models \varphi \supset \psi && \text{iff } \mathfrak{t}, w \models \varphi \text{ implies } \mathfrak{t}, w \models \psi \text{ ,}\\
&\mathfrak{t}, w \models \Box_{=}\varphi && \text{iff } \forall w' \,.\, w \mathrel{R_{=}} w' \text{ implies } \mathfrak{t}, w' \models \varphi \text{ ,}\\
&\mathfrak{t}, w \models \Box_{\neq}\varphi && \text{iff } \forall w' \,.\, w \mathrel{R_{\neq}} w' \text{ implies } \mathfrak{t}, w' \models \varphi \text{ .}
\end{aligned}
$$

A formula φ is *satisfiable* if there exists a finite data tree $\mathfrak{t}$ and a position w in $\operatorname{dom} \mathfrak{t}$ such that $\mathfrak{t}, w \models \varphi$. It is *valid* if for all finite data trees $\mathfrak{t}$ and positions w in $\operatorname{dom} \mathfrak{t}$, $\mathfrak{t}, w \models \varphi$; observe that φ is valid if and only if $\neg\varphi$ is not satisfiable, and that we can assume $w = \varepsilon$ without loss of generality by extracting the subtree of $\mathfrak{t}$ rooted by w. Note that for validity or satisfiability questions, we can assume A to be the finite set of atomic propositions appearing in φ, and work with the tag set $\Sigma \stackrel{\text{def}}{=} 2^A$.

▶ **Example 1** (Löb's Axiom)**.** Consider the following formula, known as *Löb's axiom*:

$$\Box(\Box\varphi \supset \varphi) \supset \Box\varphi \text{ ,} \tag{L}$$

which can be viewed as an induction scheme over the depth of a node: to prove that φ holds at any node, it suffices to establish that it holds at every node assuming that it holds at deeper nodes. Since we are working on finite data trees, **L** is valid: for any finite data tree $\mathfrak{t}$ and any position w in $\operatorname{dom} \mathfrak{t}$, we can show that $\mathfrak{t}, w \models \mathbf{L}$. Indeed, if we assume $\mathfrak{t}, w \models \Box(\Box\varphi \supset \varphi)$, then by induction over $h(\mathfrak{t}) - |w'|$, if $w \mathrel{R} w'$ then $\mathfrak{t}, w' \models \varphi$: this holds for any leaf w', since then $\mathfrak{t}, w' \models \Box\varphi$ vacuously, and thus $\mathfrak{t}, w' \models \varphi$; and for an inner node w', by transitivity of R all its strict descendants are also strict descendants of w, thus by induction hypothesis they satisfy φ, hence $\mathfrak{t}, w' \models \Box\varphi$ and therefore $\mathfrak{t}, w' \models \varphi$ as desired.

▶ **Example 2.** Due to the non-transitivity of $R_{\neq}$, the following variant of **L** is not valid:

$$\Box_{\neq}(\Box_{\neq}\varphi \supset \varphi) \supset \Box_{\neq}\varphi \text{ .} \tag{2}$$

A counter-model is depicted in Figure 2a in the case $\varphi \stackrel{\text{def}}{=} p$. Observe that $\mathfrak{t}, \varepsilon \not\models \Box_{\neq}p$ due to the middle node. Furthermore, $\mathfrak{t}, \varepsilon \models \Box_{\neq}(\Box_{\neq}p \supset p)$: the only node related to the root through $R_{\neq}$ is the middle node, and it satisfies $\Box_{\neq}p \supset p$ because it does not satisfy $\Box_{\neq}p$: indeed, the bottom node does not satisfy p.

▶ **Example 3.** The following, more involved formula is also valid over finite data trees, and further illustrates the interaction between $\Box_=$ and $\Box_\neq$:

$$\Box_\neq(\varphi \vee \Diamond_= \top) \supset \Diamond_\neq \varphi \vee \Box_\neq \bot \,. \tag{3}$$

Its validity relies on the finiteness assumption. Indeed, assume for the sake of contradiction that $\mathfrak{t}, w \models \Box_\neq(\varphi \vee \Diamond_= \top)$ but $\mathfrak{t}, w \not\models \Diamond_\neq \varphi$ and $\mathfrak{t}, w \not\models \Box_\neq \bot$, for some finite data tree $\mathfrak{t}$ and position w. Then there exists some w_0 in $\operatorname{dom}\mathfrak{t}$ with $w \, R_\neq \, w_0$, and $\mathfrak{t}, w_0 \not\models \varphi$. Necessarily, $\mathfrak{t}, w_0 \models \Diamond_= \top$: there exists w_1 in $\operatorname{dom}\mathfrak{t}$ such that $w_0 \, R_= \, w_1$. By cross transitivity, $w \, R_\neq \, w_1$, and we can apply the same reasoning to w_1: there exists w_2 in $\operatorname{dom}\mathfrak{t}$ such that $w_1 \, R_= \, w_2$, thus by transitivity of $R_=$ and cross transitivity, $w \, R_\neq \, w_2$, and so on and so forth. Hence there exists an infinite chain $w_0 \, R_= \, w_1 \, R_= \cdots$ of positions in $\mathfrak{t}$, which contradicts its finiteness.

Note that the previous argument describes a counter-model to (3) if we were to allow infinite data trees as models instead of only finite ones; see a counter-model in Figure 2b.

2.2.2 Data Kripke Structures

The data tree semantics of **DataGL** is actually a particular case of its semantics in terms of *data Kripke structures*. Such a structure is a tuple $\mathfrak{M} = (W, R, d, \ell)$ where W is a set of worlds, R is a binary relation over W, $d\colon W \to \mathbb{D}$ is a data labelling, and $\ell\colon W \to 2^A$ is a labelling using atomic propositions. Observe that a data tree $\mathfrak{t}$ defines such a structure with $W \stackrel{\text{def}}{=} \operatorname{dom}\mathfrak{t}$.

Given a data Kripke structure $\mathfrak{M}$, the relation R can be partitioned as $R = R_= \uplus R_\neq$ as in Equation (1), allowing to define the satisfaction relation $\mathfrak{M}, w \models \varphi$ exactly as in the case of data trees. Thus, a data Kripke structure can be seen as a Kripke structure with two relations $R_=$ and $R_\neq$. Working with this modal similarity type $\{\Diamond_=, \Diamond_\neq\}$ allows to readily apply the basic model-theoretic constructions of modal logic: satisfaction is invariant under e.g. *generated submodels* [6, Definition 2.5 and Proposition 2.6], *bounded morphisms* [6, Definition 2.12 and Proposition 2.14], and *bisimulations* [6, Definition 2.18 and Theorem 2.20].

We call a data Kripke structure a **DataGL** *model* if R is transitive irreflexive and W is finite. Given a root world w_0, the *height* of a world w is the length n of the maximal chain $w_0 \, R \, w_1 \, R \cdots R \, w_n = w$ from w_0 to w, and the *height* of $\mathfrak{M}$ the maximal height of its worlds. The semantics in terms of **DataGL** models is equivalent to that in terms of finite data trees: this is essentially due to the *tree-model property* of modal logic (see e.g. [6, Proposition 2.15], and proven via unfolding (see in the full paper for details and a reminder on bounded morphisms):

▶ **Proposition 4** (Tree-Model Property). *For any* **DataGL** *model* $\mathfrak{M}$ *and world* w_0*, there exists a finite data tree* $\mathfrak{t}_\mathfrak{M}$ *of the same height and a surjective bounded morphism* f *from* $\mathfrak{t}_\mathfrak{M}$ *to* $\mathfrak{M}$ *with* $f(\varepsilon) = w_0$*. Hence, for all* **DataGL** *formulæ* φ *and positions* w *in* $\operatorname{dom}\mathfrak{t}_\mathfrak{M}$*,* $\mathfrak{t}_\mathfrak{M}, w \models \varphi$ *if and only if* $\mathfrak{M}, f(w) \models \varphi$*.*

Since a finite data tree is a particular case of a **DataGL** model, Proposition 4 means that finite data trees and **DataGL** models can be used interchangeably.

2.2.3 Expressiveness

The logic **DataGL** is a strict fragment of **XPath**: when naming `@d` the unique data attribute of data trees, in concrete **XPath** syntax, $\Diamond_\neq \varphi$ could for instance be expressed as the node test `[./@d != ./descendant::*[`χ`]/@d]` assuming χ is the **XPath** translation of φ. More precisely, we only need the union-free fragment of $\mathbf{CoreDataXPath}^\varepsilon(\downarrow^+)$ as defined by Figueira [12],

whose satisfiability problem is EXP-complete. See in the full paper for a comparison between **DataGL** and **XPath**.

DataGL is also a strict fragment of $\mathbf{FO}^2(\sim, <)$ over data trees [7], which is the two-variables fragment of first-order logic with atomic predicates '$x \sim y$' to express that the positions at x and y have the same data values, and '$x < y$' to express that the position at y is a strict descendant of that at x. The standard translation of modal logics into first-order logics [6, Section 2.4] readily provides an equivalent $\mathbf{FO}^2(\sim, <)$ formula $\mathrm{ST}_x(\varphi)$ for any **DataGL** formula φ. To the best of our knowledge, whether satisfiability for $\mathbf{FO}^2(\sim, <)$ is decidable is open [7, 18].

3 Sequent Calculus

Developing sequent calculi for modal logics is often arduous. In order to obtain modular proof systems enjoying both cut elimination and a form of subformula property – which are desirable in order to show decidability –, advanced techniques are typically required: display logic [26], labelled calculi [20], nested sequents [8], or tree-hypersequents [21] to name a few. All these are motivated by the need to maintain additional information throughout proofs, using extra-logical means. As we are going to see, we also introduce a form of enriched sequents for **DataGL**, in the form of *histories* – which turn out to be similar to the ordered hypersequents recently employed by Indrzejczak [16] for linear-time modal logics. We define our calculus in Section 3.2, and prove it sound and complete in Sections 3.3 and 3.4. But first we present our main source of inspiration: Avron's sequent calculus for **GL** [2].

3.1 Avron's Sequent Calculus for GL

Our sequent calculus for **DataGL** is inspired by the work of Avron [2] who gave a sound and complete sequent calculus for **GL**, which does not require any extra-logical apparatus. In addition to the usual rules for Boolean connectives, Avron proposed the following sequent calculus rule ($\Box$) to deal with modalities in **GL** (the principal formula is coloured in orange):

$$\frac{\Box\Gamma, \Gamma, \Box\varphi \vdash \varphi}{\Box\Gamma \vdash \Box\varphi}\ \Box$$

As usual in classical sequent calculus, a sequent $\Gamma \vdash \Delta$ is made of two sets of formulæ: Γ (the *antecedents*) and Δ (the *consequents*) and should be interpreted as $\bigwedge_{\varphi\in\Gamma}\varphi \supset \bigvee_{\psi\in\Delta}\psi$. We simply use commas to denote set union. The notation $\Box\Gamma$ stands for the set of all $\Box\varphi$ formulæ for $\varphi \in \Gamma$, and later $\Box_=\Gamma$ will denote the set $\{\Box_=\varphi \mid \varphi \in \Gamma\}$, and similarly for $\Box_\neq$. A rule consists of a set of *premises* (the top sequents) along with a single *conclusion* (the bottom sequent). A sequent S is *derivable* from a set of sequents X if there exists a derivation with S as root and X as set of open leaves; a sequent is *provable* if it is derivable from the empty set, or equivalently if it has a *proof*, i.e. a derivation with no open leaves.

When Γ is empty, Avron's rule follows immediately from the **L** axiom: if we can prove $\Box\varphi \supset \varphi$, then we also have $\Box(\Box\varphi \supset \varphi)$ by necessitation, and $\Box\varphi$ follows by **L**. When Γ is not empty, the rule exploits the properties of $\Box$ in order to extract information from the antecedents in the conclusion sequent. Reading the rule bottom-up, the idea is that since we are assuming $\Box\Gamma$, then surely we can inductively assume Γ for any strict descendant where φ holds, but also $\Box\Gamma$ since our relation R is transitive.

▶ **Example 5** (Proof of **L** in Avron's Calculus)**.** The **L** axiom can be proven valid using ($\Box$) and the classical rules (see Figure 3 for the variants we will use later; we colour again principal formulæ in orange):

$$\dfrac{\dfrac{\dfrac{}{\Box(\Box\varphi \supset \varphi), \Box\varphi \vdash \Box\varphi, \varphi}\,\text{ax} \qquad \dfrac{}{\Box(\Box\varphi \supset \varphi), \Box\varphi, \varphi \vdash \varphi}\,\text{ax}}{\Box(\Box\varphi \supset \varphi), \Box\varphi, \Box\varphi \supset \varphi \vdash \varphi}\,\supset_L}{\Box(\Box\varphi \supset \varphi) \vdash \Box\varphi}\,\Box}{\vdash \Box(\Box\varphi \supset \varphi) \supset \Box\varphi}\,\supset_R$$

Note that the left branch of the proof could not be closed without the addition of $\Box\varphi$ among the antecedents of the premise of $(\Box)$.

The idea behind the rule $(\Box)$ applies immediately to our $\Box_=$ modality since $R_=$ is also transitive and well-founded. However, the $\Box_\neq$ modality cannot be handled in the same way because $R_\neq$ is not transitive. A sound rule for $\Box_\neq$ would be:

$$\dfrac{\Box_=\Gamma^{\neq}, \Gamma^{\neq} \vdash \varphi}{\Box_=\Gamma^{=}, \Box_\neq\Gamma^{\neq} \vdash \Box_\neq\varphi}$$

There is however no hope for this rule to yield a complete calculus. First, it fails to use in a significant way the assumptions $\Box_=\Gamma^=$. When we consider a strict descendant in the premise, we have forgotten about the position from which we came, and so we will never be able to tell when we reach a further descendant with the same datum, whether $\Gamma^=$ should hold. Second, it does not enforce tree finiteness. In that respect, note that adding an assumption $\Box_\neq\varphi$ to the antecedents of the top sequent (naively mimicking the $(\Box)$ rule) would yield an unsound rule. Both issues are solved in the following by enriching the structure of our sequents.

3.2 Sequent Calculus for DataGL

Our calculus employs sequents enriched with a *history*: those are sequences $\mathcal{H} = H_1; \dots; H_n$ of sets of modal formulæ $H_i = \Box_= H_i^=, \Box_\neq H_i^{\neq}$ called its *cells*. The length of a history $\mathcal{H}$ is denoted by $|\mathcal{H}|$. Given a history $\mathcal{H}$ and $1 \leq i \leq |\mathcal{H}|$, we always write H_i for its ith cell and $H_i^=$, $H_i^{\neq}$ for the sets such that $H_i = \Box_= H_i^=, \Box_\neq H_i^{\neq}$. If $\mathcal{H}$ is a history and H_i one of its cells, we write $\mathcal{H} \setminus H_i$ for the history obtained by removing H_i, i.e. for $H_1; \dots; H_{i-1}; H_{i+1}; \dots; H_{|\mathcal{H}|}$. The sequent calculus for **DataGL** deals with *sequents* of the form $\mathcal{H}; \Gamma \vdash \Delta$ where Γ and Δ are sets of formulæ and $\mathcal{H}$ is a history. Its rules are given in Figures 3 and 4. The calculus enjoys several desirable properties, namely that it does not contain a 'cut' rule, and that it has the *subformula property*: in any rule, the formulæ found in the premises are subformulæ of those found in the conclusion.

3.2.1 Boolean Formulæ

The rules of Figure 3 for the Boolean connectives are not surprising: the history plays no role in these rules, and when we ignore it we recover the usual sequent calculus rules for propositional classical logic. These rules are formulated in a way that avoids explicitly considering the structural rules of contraction and weakening. Note that in rules $(\supset_L)$ and $(\supset_R)$ we have chosen to keep the principal formula $\varphi \supset \psi$ in the premises of the rule (in grey in Figure 3) – this is an inessential choice that simplifies the completeness argument later, in particular Lemma 11. In fact, weakening is admissible in the calculus (meaning that adding the weakening rules does not change the set of provable sequents; see in the full paper), which means that the rules without the greyed formulæ are also admissible:

▶ **Lemma 6** (Admissibility of Weakening). *The following rules are admissible:*

$$\dfrac{\mathcal{H}; \Gamma \vdash \Delta}{\mathcal{H}; \Gamma, \varphi \vdash \Delta}\,W_L \qquad \dfrac{\mathcal{H}; \Gamma \vdash \Delta}{\mathcal{H}; \Gamma \vdash \Delta, \varphi}\,W_R$$

$$\frac{}{\mathcal{H};\Gamma,\bot\vdash\Delta}\ \bot_L \qquad \frac{}{\mathcal{H};\Gamma,\varphi\vdash\varphi,\Delta}\ \text{ax}$$

$$\frac{\mathcal{H};\Gamma,\varphi\supset\psi\vdash\varphi,\Delta \quad \mathcal{H};\Gamma,\varphi\supset\psi,\psi\vdash\Delta}{\mathcal{H};\Gamma,\varphi\supset\psi\vdash\Delta}\ \supset_L \qquad \frac{\mathcal{H};\Gamma,\varphi\vdash\varphi\supset\psi,\psi,\Delta}{\mathcal{H};\Gamma\vdash\varphi\supset\psi,\Delta}\ \supset_R$$

Figure 3 Sequent calculus for **DataGL**: classical sequent calculus rules. The principal formulæ are coloured in orange in the conclusions of the rules. The greyed formulæ $\varphi\supset\psi$ in the premises of $(\supset_L)$ and $(\supset_R)$ can be omitted; we do not use them in examples to reduce clutter.

$$\frac{\mathcal{H};\Gamma^{=},\Box_{=}\Gamma^{=},\Box_{\neq}\Gamma^{\neq},\{H_i^{\neq}\}_{1\leq i\leq|\mathcal{H}|},\Box_{=}\varphi\vdash\varphi}{\mathcal{H};\Gamma,\Box_{=}\Gamma^{=},\Box_{\neq}\Gamma^{\neq}\vdash\Box_{=}\varphi,\Delta}\ \Box_{=}$$

$$\frac{\begin{array}{c}\mathcal{H};(\Box_{=}\Gamma^{=},\Box_{\neq}\Gamma^{\neq},\Box_{\neq}\varphi);(\Gamma^{\neq},\{H_i^{\neq}\}_{1\leq i\leq|\mathcal{H}|})\vdash\varphi\\ \left\{\ \mathcal{H}\setminus H_j;(\Box_{=}\Gamma^{=},\Box_{\neq}\Gamma^{\neq},\Box_{\neq}\varphi);(\Gamma^{\neq},\{H_i^{\neq}\}_{1\leq i\leq|\mathcal{H}|,i\neq j},H_j^{=},H_j)\vdash\varphi\ \right\}_{1\leq j\leq|\mathcal{H}|}\end{array}}{\mathcal{H};\Gamma,\Box_{=}\Gamma^{=},\Box_{\neq}\Gamma^{\neq}\vdash\Box_{\neq}\varphi,\Delta}\ \Box_{\neq}$$

Figure 4 Sequent calculus for **DataGL**: modal rules. Rule $(\Box_{\neq})$ has $|\mathcal{H}|+1$ premises. The principal formulæ are coloured in orange in the conclusions of the rules. In both rules, Γ cannot contain any formula that is modal, i.e. of the form $\Box_{\star}\psi$ for $\star\in\{=,\neq\}$

3.2.2 Modal Formulæ

Before defining formally the semantics of our sequents, let us first provide an intuition for the complex rules of Figure 4 by presenting them informally from a proof search viewpoint.

Applying a modal rule bottom-up amounts to proving a sequent by considering all the possible descendants of a current (hypothetical) position in a data tree. In Avron's calculus all the important information about the current position (i.e. the modal antecedents of the conclusion sequent) could be transferred to the descendant as modal antecedents of the premise. In our case, this transfer is performed through the history. Intuitively, the history keeps track of which previous (hypothetical) positions have been visited, and of which modal formulæ were known to hold at these positions. For the same reason that Avron did not need a history, we do not need to remember past positions labelled with the same data as the current position. In fact, we only need to consider past positions labelled with mutually distinct data values: one value for the current position and one for each history cell – these values do not actually show up in the calculus, because its purpose is to establish the validity of a sequent, i.e. it simultaneously considers all possible data assignments.

The $(\Box_{=})$ rule is similar to Avron's rule, but also extracts information from the history: when we move to a strict descendant position with the same data value, it remains different from the data values of the past positions associated to history cells, and thus we know that all the $H_i^{\neq}$ formulæ hold at the new position. (Also note that this rule allows to weaken propositional formulæ, namely the parts Γ and Δ of the conclusion sequent; this is, again, to avoid considering explicit structural rules, and to facilitate proof search in our calculus.)

The $(\Box_{\neq})$ rule is the most complex one, as it not only extracts information from the history but also updates it. When moving to a strict descendant with a different data, the new data may or may not be different from the data of the previously visited positions in the history, leading to $|\mathcal{H}|+1$ premises:

- The first premise of the $(\Box_{\neq})$ rule covers the case of a totally fresh data value. In that case, we know that all the $H_i^{\neq}$ formulæ hold at the new position. We also update the history with a new cell corresponding to the position that we just left. Unsurprisingly, this cell contains the modal formulæ that were assumed about that position. It also contains $\Box_{\neq}\varphi$ as a way of mimicking well-founded inductive reasoning.
- Each of the remaining premises corresponds to the case where the new position has the same data as the position corresponding to history cell H_j. In such a case, the formulæ in $H_j^{\neq}$ are not known to hold at the new position. Instead, $H_j^{=}$ holds, as well as H_j itself, and thus there is no point in keeping a history cell for the past occurrence of the new data. As in the previous case, the history is updated with a new cell corresponding to the position we just left.

▶ **Example 7.** Let us consider again the invalid formula (2) from Example 2. Since our calculus is sound (see Theorem 9), proof search ought to fail for this formula. The first steps up to the first application of the $(\Box_{\neq})$ rule are:

$$\dfrac{\dfrac{\Box_{\neq}(\Box_{\neq}\varphi \supset \varphi), \Box_{\neq}\varphi; \Box_{\neq}\varphi \supset \varphi \vdash \varphi}{; \Box_{\neq}(\Box_{\neq}\varphi \supset \varphi) \vdash \Box_{\neq}\varphi}\,\Box_{\neq}}{; \vdash \Box_{\neq}(\Box_{\neq}\varphi \supset \varphi) \supset \Box_{\neq}\varphi}\,\supset_R$$

This creates a first history cell $H_1 \overset{\text{def}}{=} \Box_{\neq}(\Box_{\neq}\varphi \supset \varphi), \Box_{\neq}\varphi$ combining the modal formulæ of the antecedent and the principal formula. Soon after, proof search fails:

$$\dfrac{\dfrac{\dfrac{}{H_1; \Box_{\neq}\varphi; \Box_{\neq}\varphi \supset \varphi, \varphi \vdash \varphi}\,\text{ax} \qquad \text{no applicable rule}}{H_1; \vdash \Box_{\neq}\varphi, \varphi}\,\Box_{\neq} \qquad \dfrac{}{H_1; \varphi \vdash \varphi}\,\text{ax}}{H_1; \Box_{\neq}\varphi \supset \varphi \vdash \varphi}\,\supset_L$$

This second application of $(\Box_{\neq})$ to $H_1; \vdash \Box_{\neq}\varphi, \varphi$, creates a new history cell $H_2 \overset{\text{def}}{=} \Box_{\neq}\varphi$, and has two premises: the first, which assumes a fresh data, copies the formulæ under $\Box_{\neq}$ from H_1 into the antecedent; the second assumes we have encountered the same data value as in the position remembered through H_1, thus copies the formulæ under $\Box_{=}$ in H_1 (there are none) into the antecedent, but also extracts H_1 itself from the history and puts it in the antecedent. Note that this search was deterministic: there were never any alternative applicable rule, hence formula (2) is unprovable.

3.3 Soundness

We now formally define the semantics of our sequents, and establish that the calculus is sound. An *annotated sequent* $(\mathcal{H}; \Gamma \vdash \Delta)^d$ is a sequent $\mathcal{H}; \Gamma \vdash \Delta$ together with a *data assignment* d, which is an injective function from $\{0, \dots, |\mathcal{H}|\}$ to $\mathbb{D}$. We write $d_0, \dots, d_{|\mathcal{H}|}$ for those distinct data values: the data value d_0 is understood as being associated to the bare sequent $\Gamma \vdash \Delta$, while each d_i for $0 < i \leq |\mathcal{H}|$ is associated to the cell H_i.

▶ **Definition 8** (Sequent Satisfiability and Validity)**.** A finite data tree $\mathfrak{t}$ *satisfies* an annotated sequent $(\mathcal{H}; \Gamma \vdash \Delta)^d$ at a position w in $\operatorname{dom}\mathfrak{t}$ if and only if the following conditions together imply $\mathfrak{t}, w \models \varphi$ for some $\varphi \in \Delta$:

(a) $\mathfrak{t}, w \models \varphi$ for all $\varphi \in \Gamma$,
(b) $d(w) = d_0$,
(c) $\mathfrak{t}, w' \models \varphi$ for all $1 \leq i \leq |\mathcal{H}|$, all $\Box_{=}\varphi \in H_i^{=}$, and all w' with $w \mathrel{R} w'$ and $d(w') = d_i$, and
(d) $\mathfrak{t}, w' \models \varphi$ for all $1 \leq i \leq |\mathcal{H}|$, all $\Box_{\neq}\varphi \in H_i^{\neq}$, and all w' with $w \mathrel{R} w'$ and $d(w') \neq d_i$.

An annotated sequent is *valid* if and only if it is satisfied by all finite data trees at all positions. A finite data tree $\mathfrak{t}$ *satisfies* a sequent S at a position w, written $\mathfrak{t}, w \models S$, if it satisfies some annotation of S at w. A sequent is *valid* if all its annotations are valid.

Note that the validity of one annotation of a sequent is equivalent to the validity of all of its annotations, because any two annotations are related by a bijective renaming of data values, and satisfaction of a formula is invariant under such renamings. We prove soundness in in the full paper by checking that the rules preserve validity:

▶ **Theorem 9** (Soundness). *The sequent calculus for* **DataGL** *is sound, i.e. all provable sequents are valid.*

3.4 Completeness

We now turn to proving that our sequent calculus is complete: φ is valid in **DataGL** if and only if $\vdash \varphi$ is provable. We show more generally that our calculus is complete with respect to sequent validity:

▶ **Theorem 10** (Completeness). *The sequent calculus for* **DataGL** *is complete, i.e. all valid sequents are provable.*

This result is obtained by constructing for any invalid sequent an appropriate *canonical* (counter-)model, which is a **DataGL** model. More specifically, we follow Avron [2] in building a model based on unprovable saturated sequents, rather than a model based on saturated sets of formulæ as in the Hilbert-style approach [6]. This is not a minor difference: it allows us to obtain a canonical model that enjoys irreflexivity and well-foundedness, two properties that are not directly obtained in the Hilbert-style approach.

Let us call a sequent $\mathcal{H}; \Gamma \vdash \Delta$ *saturated* if $\varphi \supset \psi \in \Gamma$ implies $\varphi \in \Delta$ or $\psi \in \Gamma$, and $\varphi \supset \psi \in \Delta$ implies $\varphi \in \Gamma$ and $\psi \in \Delta$. We can restrict ourselves to work with saturated sequents without loss of generality:

▶ **Lemma 11** (Saturation Lemma). *For any unprovable sequent $S = \mathcal{H}; \Gamma \vdash \Delta$ there exists an unprovable sequent $S' = \mathcal{H}; \Gamma' \vdash \Delta'$ using only subformulæ of S, which is saturated and such that $\Gamma \subseteq \Gamma'$, $\Delta \subseteq \Delta'$. Furthermore if $\mathfrak{t}, w \models S$ then $\mathfrak{t}, w \models S'$.*

Proof sketch. The total size of all formulæ for which the saturation condition fails can be decreased by repeatedly applying the rules of Figure 3 bottom-up. This process yields a set of saturated sequents X, and a simple inspection of the rules shows that for all the sequents S' in X the formulæ initially present in Γ (resp. Δ) are still present, that S' only contains subformulæ of S, and that $\mathfrak{t}, w \models S$ implies $\mathfrak{t}, w \models S'$. Since S was unprovable, X contains at least one unprovable sequent. ◀

We shall build our canonical model based on annotated saturated sequents. In order to obtain a finite model, we restrict our sequents to only contain (sub)formulæ among a finite set, and we forbid duplicates in histories. This, in turn, allows us to bound the number of possible data assignments. In the remainder of this section, let us fix a finite set of formulæ $\mathcal{F}$ that is closed under taking subformulæ. Since we want to exhibit a counter-model, we can indeed work here with $\mathbb{D} \stackrel{\text{def}}{=} \mathbb{N}$ without loss of generality.

▶ **Definition 12** (Canonical Sequents). A *canonical sequent* $(\mathcal{H}; \Gamma \vdash \Delta)^d$ over $\mathcal{F}$ is unprovable saturated annotated sequent such that its formulæ belong to $\mathcal{F}$, $H_i \neq H_j$ for $1 \leq i \neq j \leq |\mathcal{H}|$, and $0 \leq d_i \leq 2^{|\mathcal{F}|}$ for every $0 \leq i \leq |\mathcal{H}|$.

Given a sequent $S = \mathcal{H}; \Gamma \vdash \Delta$, we write $S_i \stackrel{\text{def}}{=} H_i$ for $1 \leq i \leq |\mathcal{H}|$ for its history cells and $S_0 \stackrel{\text{def}}{=} \{\Box_\star \varphi \mid \Box_\star \varphi \in \Gamma, \star \in \{=, \neq\}\}$ for the set of modal formulæ found in Γ. We shall refer to the sets $(S_i)_{i \geq 0}$ as the *cells* of S.

▶ **Definition 13** (Canonical Relation). Given two sequents $S = \mathcal{H}; \Gamma \vdash \Delta$ and $S' = \mathcal{H}'; \Gamma' \vdash \Delta'$, S *embeds modally* into S', written $S \sqsubseteq S'$, if there exists an injective function $f : \{0, \ldots, |\mathcal{H}|\} \to \{0, \ldots, |\mathcal{H}'|\}$ such that $S_i \subseteq S'_{f(i)}$ for all $0 \leq i \leq |\mathcal{H}|$.

Given annotations d and d' for S and S', we define $S \; R^{\mathfrak{c}} \; S'$ to hold whenever $S \sqsubseteq S'$ and

- **(i)** for all $0 \leq i \leq |\mathcal{H}|$, $d_i = d'_{f(i)}$,
- **(ii)** $\varphi \in \Gamma'$ whenever $\Box_= \varphi \in S_i$ with $f(i) = 0$,
- **(iii)** $\psi \in \Gamma'$ whenever $\Box_{\neq} \psi \in S_i$ with $f(i) \neq 0$, and
- **(iv)** there exists some formula $\Box_\star \varphi \in (\Delta \setminus \Gamma) \cap S'_{f(0)}$ for $\star \in \{=, \neq\}$; that formula is called the *witness* associated to $S \; R^{\mathfrak{c}} \; S'$.

▶ **Lemma 14** (Transitivity and Irreflexivity). *The relation $R^{\mathfrak{c}}$ is transitive and irreflexive.*

Proof. Transitivity is obvious for $\sqsubseteq$ and conditions (i)–(iii). For condition (iv), it comes from the fact that witnesses are propagated by $\sqsubseteq$. Indeed, if $S \; R^{\mathfrak{c}} \; S' \; R^{\mathfrak{c}} \; S''$, then the witness $\Box_\star \varphi$ for $S \; R^{\mathfrak{c}} \; S'$ belongs to $\Delta \setminus \Gamma$ and to $S'_{f(i)}$ such that $d'_i = d_0$. Thus by condition (i) it also belongs to S''_j for $d''_j = d'_i = d_0$. Regarding irreflexivity, if $S \; R^{\mathfrak{c}} \; S$ then the witness would have to belong to $(\Delta \setminus \Gamma) \cap S'_0$ since $d_0 = d'_0$ (and all other data values are distinct from d_0), but that set is empty since $S'_0 \subseteq \Gamma$. ◀

Given a finite $\mathcal{F}$, observe that the set $C(\mathcal{F})$ of canonical sequents over $\mathcal{F}$ is also finite.

▶ **Definition 15** (Canonical Structure). The *canonical structure* over $\mathcal{F}$ is the data Kripke structure $\mathfrak{C}(\mathcal{F}) \stackrel{\text{def}}{=} (C(\mathcal{F}), R^{\mathfrak{c}}, d^{\mathfrak{c}}, \ell^{\mathfrak{c}})$ over canonical sequents. The data label of a sequent $S = (\mathcal{H}; \Gamma \vdash \Delta)^d$ is $d^{\mathfrak{c}}(S) \stackrel{\text{def}}{=} d_0$ and its propositional label is the set of atomic propositions found in Γ, i.e. $\ell^{\mathfrak{c}}(S) \stackrel{\text{def}}{=} \{p \in A \mid p \in \Gamma\}$.

By Lemma 14 and the finiteness of $\mathfrak{C}(\mathcal{F})$, it is a **DataGL** model. Hence Proposition 4 can be invoked to show the existence of a finite data tree satisfying the same sequents. The following lemma shows that $\mathfrak{C}(\mathcal{F})$ provides counter-models to validity in the sense of Definition 8 (see in the full paper for a proof):

▶ **Lemma 16** (Falsification Lemma). *For any canonical sequent $S = (\mathcal{H}; \Gamma \vdash \Delta)^d$ we have:*

- $\mathfrak{C}(\mathcal{F}), S \models \varphi$ *for all* $\varphi \in \Gamma$*;*
- $\mathfrak{C}(\mathcal{F}), S' \models \varphi$ *for all* $1 \leq i \leq |\mathcal{H}|$*, all* $\Box_= \varphi \in H_i$*, and all* S' *with* $S \; R^{\mathfrak{c}} \; S'$ *and* $d^{\mathfrak{c}}(S') = d_i$*;*
- $\mathfrak{C}(\mathcal{F}), S' \models \varphi$ *for all* $1 \leq i \leq |\mathcal{H}|$*, all* $\Box_{\neq} \varphi \in H_i$*, and all* S' *with* $S \; R^{\mathfrak{c}} \; S'$ *and* $d^{\mathfrak{c}}(S') \neq d_i$*;*
- $\mathfrak{C}(\mathcal{F}), S \not\models \varphi$ *for all* $\varphi \in \Delta$*.*

We can finally establish our result:

Proof of Theorem 10. Consider a sequent $\mathcal{H}; \Gamma \vdash \Delta$ that is unprovable. We can assume without loss of generality that it has no duplicate cell, as we can always add dummy formulæ to differentiate between identical cells, without making the sequent provable. Define $\mathcal{F}$ as its set of subformulæ. By Lemma 11 we obtain an unprovable saturated sequent $\mathcal{H}; \Gamma, \Gamma' \vdash \Delta, \Delta'$. This sequent can be annotated to obtain a canonical sequent, and by Lemma 16 we have a counter-model of that sequent, which is also a counter-model of $\mathcal{H}; \Gamma \vdash \Delta$. ◀

4 Proof Search

In this section we analyse further the structure of our proof system, deriving properties that are useful for proof search. We first discuss a straightforward complete proof search strategy in Section 4.1. In spite of its simplicity, it yields a polynomial bound on the depth of proofs (see Section 4.2) and therefore a PSPACE upper bound on **DataGL** validity, which is optimal (see Section 4.3).

4.1 Proof Search Strategy

Proof search can be understood intuitively as a game between two players Prover and Spoiler with sequents as positions. Given a sequent S, Prover first chooses an applicable rule, i.e. a rule whose conclusion matches S. Spoiler then chooses the new current sequent among the premises of the rule application. Any player with no possible move loses: Prover if no rule is applicable, and Spoiler if there are no premises, as with rules (ax) and $(\bot_L)$; furthermore Spoiler wins if the play is infinite. A sequent is valid if and only if Prover has a winning strategy in this game.

When several rules are applicable, which one should Prover select?

- Clearly, if either (ax) or $(\bot_L)$ is applicable, then she should pick it since she wins immediately.
- Furthermore, the rules $(\supset_L)$ and $(\supset_R)$ are *invertible*, i.e. their premises are provable if and only if their conclusion is provable: one direction is immediate since the premises allow to derive the conclusion, and conversely we can appeal to weakening (recall Lemma 6) to show the provability of the premises from that of the conclusion. An obvious complete proof search strategy is then to apply $(\supset_L)$ and $(\supset_R)$ eagerly, decomposing any Boolean formula that is not yet decomposed.
- After this phase, the only hope to prove a sequent is to use a modal rule: Prover has to select one principal modal formula on the right of the sequent and apply the corresponding $(\Box_=)$ or $(\Box_\neq)$ rule.

Although the obtained strategy only makes essential choices, it may *a priori* diverge: each application of a modal rule imports new Boolean formulas into the antecedent, which may yield new modal rules, and so on and so forth. It turns out, however, that the 'GL component' of our modal rules allows us to derive a small bound on the length of branches that should be considered in proof search attempts.

4.2 Small Proof Property

The *depth* of a proof Π is the maximal number of rule applications in any of its branches, in other words the maximal length of a play in the proof search game. The *size* of a proof $|\Pi|$ is the total number of rule applications in the proof tree. A proof is *minimal* if no other proof of its conclusion sequent is of strictly smaller size. Note that a minimal proof must necessarily apply the (ax) and $(\bot_L)$ rules as soon as possible; it also cannot decompose twice the same Boolean formula between two applications of a modal rule $(\Box_=)$ or $(\Box_\neq)$.

Inspecting the rules of our calculus, it appears that a cell in the history – including S_0, the modal part of the current antecedent – may be displaced, perhaps moved to the antecedent of the sequent, be enriched with new modal formulæ, but can never be lost:

▶ **Lemma 17.** *Let Π be a derivation of a sequent S. We have $S \sqsubseteq S'$ for any sequent S' occurring in Π.*

▶ **Proposition 18** (Bounded ($\Box_{\neq}$) Applications). *Let Π be a minimal proof. For any formula $\Box_{\neq}\psi$, each branch of Π contains at most three applications of the ($\Box_{\neq}$) rule with $\Box_{\neq}\psi$ as principal formula.*

Proof. Let Π be a minimal proof of $\mathcal{H}^0;\Gamma^0 \vdash \Delta^0$. Consider a branch of Π that contains three applications of ($\Box_{\neq}$) to the same formula $\Box_{\neq}\psi$. Let $S^k = (\mathcal{H}^k;\Gamma^k \vdash \Delta^k)_{0\leq k\leq B}$ be the (unannotated) sequents along that branch, of length B, and let $p < q < r$ be the indices of the conclusion sequents of the three successive applications of ($\Box_{\neq}$) with $\Box_{\neq}\psi$ as principal formula. We shall establish that the axiom rule applies after the third rule application, thus a fourth application is impossible in a minimal proof.

The first application of ($\Box_{\neq}$) with $\Box_{\neq}\psi$ as principal formula introduces (in its premise sequent) a new history cell containing $\Box_{\neq}\psi$. By Lemma 17, for all $k > p$, there exists $0 \leq i_k \leq |\mathcal{H}^k|$ such that $\Box_{\neq}\psi \in S^k_{i_k}$.

By minimality of Π we know that the axiom rule does not apply when the second ($\Box_{\neq}$) rule with $\Box_{\neq}\psi$ as principal formula is performed, thus $\Box_{\neq}\psi \notin \Gamma^q$ and $i_q > 0$. For the same reason we also have $\psi \notin \Gamma^{q+1}$. Thus the premise of this second rule application cannot be the first premise of the ($\Box_{\neq}$) rule, for otherwise ψ would belong to Γ^{q+1}. For the same reason, it cannot correspond to one of the other premises with $j \neq i_q$. Hence, it must be the premise with $j = i_q$, and S^{q+1} contains two distinct cells containing $\Box_{\neq}\psi$, namely the antecedent and the freshly created cell. By Lemma 17 again, it follows that for all $k > q$, $\Box_{\neq}\psi \in S^k_{j_k}$ for some cell index $0 \leq j_k \neq i_k \leq |\mathcal{H}^k|$.

For the third rule application, the same argument applies, but this time all the premises of the rule contain ψ in their antecedent Γ^{r+1}, i.e. the axiom rule is applicable immediately after the third ($\Box_{\neq}$) application. ◀

Note that we can create a new history cell only when we apply a modal rule ($\Box_{\neq}$), thus Proposition 18 indirectly provides a bound on the length of the history in minimal proofs.

▶ **Proposition 19** (Bounded ($\Box_{=}$) Applications). *Let Π be a minimal proof in which history lengths are bounded by h. For any formula $\Box_{=}\varphi$, all branches of Π contain at most $h+1$ applications of the ($\Box_{=}$) rule with $\Box_{=}\varphi$ as principal formula.*

Proof sketch. Each application of ($\Box_{=}$) can only occur with conclusion sequents where $\Box_{=}\varphi$ does not appear in the antecedent Γ, or the axiom rule could be applied instead. Thus its premise sequent has one more cell containing $\Box_{=}\varphi$ than its conclusion sequent, namely in the antecedent. By Lemma 17, we can only repeat this operation $h+1$ times before being forced to see $\Box_{=}\varphi$ in the antecedent Γ. ◀

From the previous two results we easily obtain the following statement, which also takes Boolean rules into account to obtain a polynomial bound on the depth of minimal proofs.

▶ **Theorem 20** (Polynomial Proof Depth). *Let $S = \mathcal{H};\Gamma \vdash \Delta$ be a sequent, $m^{\neq}$ (resp. $m^{=}$) be the number of distinct $\Box_{\neq}$ subformulæ (resp. $\Box_{=}$ subformulæ) occurring in S, and p be the number of other subformulæ. If S is provable, then it has a proof of depth at most $(3m^{\neq} + m^{=}(3m^{\neq}+1)+1)(p+1)+1$.*

Proof. By Proposition 18 and the subformula property, the number of ($\Box_{\neq}$) rule applications along any branch is bounded by $3m^{\neq}$, which is also a bound on the length of histories in minimal proofs. By Proposition 19 and the subformula property, the total number of modal rule applications is thus bounded by $3m^{\neq} + m^{=}(3m^{\neq}+1)$. In between any two modal rules we can apply at most p Boolean rules without introducing the same formula twice, and possibly conclude by an axiom. ◀

4.3 Computational Complexity

The polynomial bound of Theorem 20 on the length of branches in minimal proofs yields a proof search algorithm working in alternating polynomial time. The algorithm implements the proof search game with Prover as existential player and Spoiler as universal player. Actually, Prover can follow the strategy of Section 4.1 without loss of generality nor of efficiency. Since our calculus is sound and complete for **DataGL**, and AP = PSPACE [10], it yields a PSPACE upper bound for the validity problem for **DataGL**. Because **DataGL** is closed under negation and PSPACE under complement, the same bound holds for the satisfiability problem. We show that this is actually a tight bound.

▶ **Theorem 21.** *The validity and satisfiability problems for* **DataGL** *are* PSPACE*-complete.*

The lower bound is obtained by reducing the satisfiability problem for **GL** to its **DataGL** counterpart. PSPACE-hardness follows since **GL** validity is known to be PSPACE-hard [9, Theorem 7]. We can indeed think of a finite tree without data as a finite data tree where all the nodes share the same datum, and in such a tree $\Box_=$ behaves exactly as the **GL** modality $\Box$. Given a **GL** formula φ we define the **DataGL** formula $\lfloor\varphi\rfloor$ by substituting $\Box_=$ for $\Box$:

$$\lfloor p\rfloor \stackrel{\text{def}}{=} p\,, \qquad \lfloor\bot\rfloor \stackrel{\text{def}}{=} \bot\,, \qquad \lfloor\varphi\supset\psi\rfloor \stackrel{\text{def}}{=} \lfloor\varphi\rfloor\supset\lfloor\psi\rfloor\,, \qquad \lfloor\Box\varphi\rfloor \stackrel{\text{def}}{=} \Box_=\lfloor\varphi\rfloor\,.$$

▶ **Claim 22.** *For any* **GL** *formula φ, φ is satisfiable in* **GL** *if and only if $\lfloor\varphi\rfloor\wedge\Box_{\neq}\bot$ is satisfiable in* **DataGL***.*

Proof sketch. The formula $\Box_{\neq}\bot$ ensures that a single data value is used throughout its models. Given such a finite single-data data tree $\mathfrak{t}$, a straightforward structural induction over φ establishes that $\mathfrak{t}, w \models \lfloor\varphi\rfloor$ if and only if $\lfloor\mathfrak{t}\rfloor, w \models \varphi$, where $\lfloor\mathfrak{t}\rfloor$ is the tree obtained by erasing all data information from $\mathfrak{t}$. ◀

A final observation about proof search is that the polynomial bound of Theorem 20 on the depth also applies to *failed* searches that follow the strategy of Section 4.1. This is exploited by Lunel [19] to extract small counter-models from failed proof attempts when focusing on saturated sequents (the completeness proof in Section 3.4 is indeed essentially based on proof search): if a sequent is unprovable, then it has a counter-model of polynomial height. Hence **DataGL** has a *strong finite model property*. It yields a different proof of the PSPACE upper bound of Theorem 21 when combined with Proposition 6.7 of [12], though with a rather less concrete algorithm than proof search.

5 Concluding Remarks

The sequent calculus for **DataGL** is to the best of our knowledge the first instance of a proof system for a data-aware logic on finite data trees. It provides an optimal proof search algorithm to establish the validity of **DataGL** formulæ, with a PSPACE complexity.

The logic **DataGL** is still a rather small syntactic fragment of **CoreDataXPath**. There are two natural directions for extending our proof system towards full **CoreDataXPath**:

- one is allowing path expressions as in **CoreDataXPath**($\downarrow^+$), which is EXP-complete [12],
- the other is to add other navigational axes, starting with the ancestor axis $\uparrow^+$, as in the non primitive recursive **CoreDataXPath**($\uparrow^+,\downarrow^+$) [15].

On both accounts, the use of sequents enriched with histories is a promising starting point.

From a proof theory perspective, two lines of inquiry seem interesting. The first would be to develop a cut elimination procedure for our sequent calculus; by completeness of the

calculus, the (cut) rule is admissible, but this is a semantic proof rather than a syntactic one. The second is to consider an extension of **DataGL** where histories are integrated as logical connectives in the syntax instead of being a mere extra-logical mechanism; this might help in designing Hilbert-style axiomatisations for data logics, along the same lines as the recent work of Abriola et al. [1] on XPath equipped with the immediate child relation.

Acknowledgements. This work benefited greatly from discussions with Diego Figueira and Luc Segoufin. We thank the anonymous reviewers for their helpful comments.

References

1 Sergio Abriola, María Emilia Descotte, Raul Fervari, and Santiago Figueira. Axiomatizations for downward XPath on data trees. Preprint, 2016. URL: http://arxiv.org/abs/1605.04271.

2 Arnon Avron. On modal systems having arithmetical interpretations. *J. Symb. Log.*, 49(3):935–942, 1984. doi:10.2307/2274147.

3 Michael Benedikt, Wenfei Fan, and Floris Geerts. XPath satisfiability in the presence of DTDs. *Journal of the ACM*, 55(2):1–79, 2008. doi:10.1145/1346330.1346333.

4 Michael Benedikt, Wenfei Fan, and Gabriel Kuper. Structural properties of XPath fragments. *Theor. Comput. Sci.*, 336(1):3–31, 2005. doi:10.1016/j.tcs.2004.10.030.

5 Michael Benedikt and Christoph Koch. XPath leashed. *ACM Computing Surveys*, 41(1):3:1–3:54, 2009. doi:10.1145/1456650.1456653.

6 Patrick Blackburn, Marteen de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

7 Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):1–48, 2009. doi:10.1145/1516512.1516515.

8 Kai Brünnler and Lutz Straßburger. Modular sequent systems for modal logic. In *Tableaux 2009*, volume 5607 of *LNCS*, pages 152–166. Springer, 2009. doi:10.1007/978-3-642-02716-1_12.

9 A. V. Chagrov and M. N. Rybakov. How many variables does one need to prove PSPACE-hardness of modal logics? In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *AiML 2002*, pages 71–82. King's College Publications, 2003.

10 Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.

11 Diego Figueira. Alternating register automata on finite words and trees. *Logic. Meth. in Comput. Sci.*, 8(1):1–43, 2012. doi:10.2168/LMCS-8(1:22)2012.

12 Diego Figueira. Decidability of downward XPath. *ACM Trans. Comput. Logic*, 13(4):1–40, 2012. doi:10.1145/2362355.2362362.

13 Diego Figueira. On XPath with transitive axes and data tests. In *PODS 2013*, pages 249–260. ACM, 2013. doi:10.1145/2463664.2463675.

14 Diego Figueira, Santiago Figueira, and Carlos Areces. Model theory of XPath on data trees. Part I: bisimulation and characterization. *J. Artif. Intell. Res.*, 53(1):271–314, 2015. doi:10.1613/jair.4658.

15 Diego Figueira and Luc Segoufin. Bottom-up automata on data trees and vertical XPath. In *STACS 2011*, volume 9 of *LIPIcs*, pages 93–104. LZI, 2011. doi:10.4230/LIPIcs.STACS.2011.93.

16 Andrzej Indrzejczak. Linear time in hypersequent framework. *Bull. Symb. Logic*, 22(1):121–144, 2016. doi:10.1017/bsl.2016.2.

17 M. Jurdziński and R. Lazić. Alternating automata on data trees and XPath satisfiability. *ACM Trans. Comput. Logic*, 12(3):1–21, 2011. doi:10.1145/1929954.1929956.

18 Emmanuel Kieroński and Lidia Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *LICS 2009*, pages 123–132, 2009. doi:10.1109/LICS.2009.39.

19 Simon Lunel. Systèmes de preuves pour logiques modales à données. Mémoire de Master, LMFI, Université Paris-Diderot, September 2015.

20 Sara Negri. Proof analysis in modal logic. *J. Phil. Log.*, 34(5–6):507–544, 2005. doi:10.1007/s10992-005-2267-3.

21 Francesca Poggiolesi. The method of tree-hypersequents for modal propositional logic. In David Makinson, Jacek Malinowski, and Heinrich Wansing, editors, *Towards Mathematical Philosophy*, volume 28 of *Trends in Logic*, pages 31–51. Springer, 2009. doi:10.1007/978-1-4020-9084-4_3.

22 Jonathan Robie, Don Chamberlin, Michael Dyck, and John Snelson. XML Path Language (XPath) 3.0. W3C Recommendation, World Wide Web Consortium (W3C), 2014. URL: http://www.w3.org/TR/xpath-30/.

23 Balder ten Cate, Gaëlle Fontaine, and Tadeusz Litak. Some modal aspects of XPath. *J. Appl. Non-Classical Log.*, 20(3):139–171, 2010. doi:10.3166/jancl.20.139-171.

24 Balder ten Cate, Tadeusz Litak, and Maarten Marx. Complete axiomatizations for XPath fragments. *J. Appl. Logic*, 8(2):153–172, 2010. doi:10.1016/j.jal.2009.09.002.

25 Balder ten Cate and Maarten Marx. Axiomatizing the logical core of XPath 2.0. *Theor. Comput. Sys.*, 44(4):561–589, 2009. doi:10.1007/s00224-008-9151-9.

26 Heinrich Wansing. Sequent calculi for normal modal propositional logics. *J. Logic Comput.*, 4(2):125–142, 1994. doi:10.1093/logcom/4.2.125.

Axiomatizations for Propositional and Modal Team Logic

Martin Lück

Leibniz Universität Hannover, Germany
lueck@thi.uni-hannover.de

Abstract

A framework is developed that extends Hilbert-style proof systems for propositional and modal logics to comprehend their team-based counterparts. The method is applied to classical propositional logic and the modal logic K. Complete axiomatizations for their team-based extensions, propositional team logic PTL and modal team logic MTL, are presented.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases team logic, propositional team logic, modal team logic, proof system, axiomatization

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.33

1 Introduction

Propositional and modal logics, while their history goes back to ancient philosophers, have assumed an outstanding role in the age of modern computer science, with plentiful applications in software verification, modeling, artificial intelligence, and protocol design. An important property of a logical framework is *completeness*, i.e., that the act of mechanical reasoning can effectively be done by a computer.

A recent extension of classical logics is the generalization to *team semantics*, i.e., formulas are evaluated on whole sets of assignments. So-called *team based logics* allow a more sophisticated expression of facts that regard multiple states of a system simultaneously as well as their internal relationship towards each other. The concept of team logic originated from the idea of quantifier dependence and independence. The question was simple and is long-known in linguistics: How can the statement

For every x there is $y(x)$, and for every u there is $v(u)$ such that P(x,y,u,v)

be formalized? The fact that v should only depend on u cannot be expressed with first-order quantifiers. Some suggestions were the *independence-friendly logic* $\mathcal{IF}$ by Hintikka and Sandu [7] or the *dependence logic* $\mathcal{D}$ by Väänänen [15]. Hodges found that a compositional semantics of $\mathcal{IF}$ can be formulated with the concept of teams [8], which was adapted by Väänänen [14, 15] together with an *atom of dependence*, written $=(x, y)$ or $\mathsf{dep}(x, y)$.

Beside Väänänen's dependence atom a variety of atomic formulas solely for the reasoning in teams were introduced. Galliani and others found a connection to database theory; they defined common constraints like *independence* $\perp$, *inclusion* $\subseteq$ and *exclusion* $|$ in the framework of team semantics [2, 4]. Beside first-order logic, all these atoms were also adapted for modal logic $\mathcal{ML}$ [14] and recently propositional logic $\mathcal{PL}$ [16].

As for any logic system, the question of axiomatizability arose. After all, team logics enable reasoning about sets of valuations, and predicate logic with set quantifiers ($\mathcal{SO}$) is not axiomatizable. An important connection to team logic was found in the sense that dependence

© Martin Lück;
licensed under Creative Commons License CC-BY

logic $\mathcal{D}$ is as powerful as existential second-order logic $\mathcal{SO}(\exists)$ [15], and that its extension $\mathcal{TL}$ (where a semantical negation $\sim$ is provided) is even equivalent to full second-order logic $\mathcal{SO}$ [10]; therefore both are non-axiomatizable. Later Kontinen and Väänänen showed that there is a partial axiomatization in the sense that $\mathcal{FO}$ consequences of $\mathcal{D}$ formulas are derivable [11]. For many weaker team logics the question of axiomatizability is open. Exceptions are certain fragments of propositional and modal team logic. They were axiomatized by Sano and Virtema [13] and Yang [16], but these solutions rely on the absence of Boolean negation.

Contribution

In this paper complete axiomatizations of $\mathcal{PTL}$ and $\mathcal{MTL}$ are given, the full propositional and modal team logics. A crucial step in the completeness proof is the fact that $\mathcal{PTL}$ is not more expressive than a (team semantical) Boolean combination of classical $\mathcal{PL}$ formulas, in symbols $\mathcal{PTL} \equiv \mathcal{B}(\mathcal{PL})$. In the modal case analogously $\mathcal{MTL} \equiv \mathcal{B}(\mathcal{ML})$ holds. For a similar application to first-order logic see the technical report [12].

The paper is built as follows: After reminding the reader of several foundational definitions (Section 2), complete axiomatizations for the Boolean closures $\mathcal{B}(\mathcal{PL})$ and $\mathcal{B}(\mathcal{ML})$ are presented (Section 3). The collapses from $\mathcal{PTL}$ and $\mathcal{MTL}$ to $\mathcal{B}(\mathcal{PL})$ and $\mathcal{B}(\mathcal{ML})$ are then proven step-wise by axiomatizing the elimination of splitting (Section 4) and modalities (Section 5).

2 Preliminaries

If in the following A is a set, then $\mathfrak{P}(A)$ refers to its power set. The notation $[n]$ will be used for the set $\{1, \dots, n\}$, assuming $n \in \mathbb{N}$.

We define a *logic* as a triple $\mathcal{L} = (\Phi, \mathfrak{A}, \models)$. The component Φ is a countable set consisting of finite words over some alphabet Σ, the so-called *formulas* of $\mathcal{L}$. The set $\mathfrak{A}$ contains possible *valuations* of formulas in Φ, and the binary relation $\models$ is the *truth* or *satisfaction relation* between $\mathfrak{A}$ and Φ. To distinguish between different satisfaction relations we sometimes write $\models_{\mathcal{L}}$. We use the same symbol for the *entailment relation*, $\varphi \models \psi$ meaning that $\forall A \in \mathfrak{A} : A \models \varphi$ implies $A \models \psi$. These relations are as usual generalized to sets, $A \models \Phi$ meaning $\forall \varphi \in \Phi : A \models \varphi$, and $\Phi \models \psi$ meaning that $\forall A \in \mathfrak{A} : A \models \Phi$ implies $A \models \psi$.

The reader is assumed to be familiar with the foundations of classical propositional and modal logics. We define classical propositional logic via a countable set $\mathcal{PS} := \{x_1, x_2, \dots\}$ of atomic propositional statements and the connectives $\to$ and $\neg$. Truth $\top$ and falsum $\bot$ are defined as $(x_1 \to x_1)$ and $\neg\top$, respectively. On top of propositional logic, modal logic is defined with the additional unary modality $\Box$ with the standard Kripke semantics.

2.1 Team logics

Let $\mathcal{L}$ be a logic. We introduce two new operators to $\mathcal{L}$: The unary *strong negation* $\sim$ and the binary *material implication* $\mathrel{-\!\!\!\rightarrow}$ (under the assumption that they are not symbols in formulas of $\mathcal{L}$). The logic $\mathcal{B}(\mathcal{L})$ is the *Boolean closure* of $\mathcal{L}$ and is defined by the following grammar, where α stands for any $\mathcal{L}$-formula: $\varphi ::= \alpha \mid {\sim}\varphi \mid (\varphi \mathrel{-\!\!\!\rightarrow} \varphi)$. Note that in particular any atom of the logic $\mathcal{L}$ is an atom of $\mathcal{B}(\mathcal{L})$, but connecting $\mathcal{L}$-formulas with $\sim$ or $\mathrel{-\!\!\!\rightarrow}$ always yields formulas not in $\mathcal{L}$. We further use the symbol $\perp\!\!\!\perp$ (*strong falsum*), $\perp\!\!\!\perp := {\sim}(\psi \mathrel{-\!\!\!\rightarrow} \psi)$, and the abbreviations $(\varphi \mathbin{⍰} \psi) := ({\sim}\varphi \mathrel{-\!\!\!\rightarrow} \psi)$, $(\varphi \mathbin{⍟} \psi) := {\sim}(\varphi \mathrel{-\!\!\!\rightarrow} {\sim}\psi)$ and $(\varphi \mathrel{-\!\!\!\leftrightarrow} \psi) := (\varphi \mathrel{-\!\!\!\rightarrow} \psi) \mathbin{⍟} (\psi \mathrel{-\!\!\!\rightarrow} \varphi)$. The semantics of $\mathcal{B}(\mathcal{L})$ extend $\mathcal{L}$ by, given some valuation $A \in \mathfrak{A}$ of $\mathcal{L}$, as follows: $A \models {\sim}\varphi \Leftrightarrow A \not\models \varphi$ and $A \models \varphi \mathrel{-\!\!\!\rightarrow} \psi \Leftrightarrow A \not\models \varphi$ or $A \models \psi$.

The next operator introduced in team logic is the binary operator $-\!\circ$, called *linear implication*, similar as in Väänänen's first-order team logic $\mathcal{TL}$ [15]. Assume that the logic $\mathcal{L} = (\Phi, \mathfrak{A}, \vDash)$ has a *splitting relation* $\sigma_{\mathcal{L}} \subseteq \mathfrak{A}^3$. If $(A, B, C) \in \sigma_{\mathcal{L}}$ then we say that (B, C) is a *splitting* or *division* of A. The semantics is that $A \vDash \varphi -\!\circ \psi$ if for all (B, C) with $(A, B, C) \in \sigma_{\mathcal{L}}$ it holds $B \nvDash \varphi$ or $C \vDash \psi$. Abbreviate $\varphi \otimes \psi := {\sim}(\varphi -\!\circ {\sim}\psi)$. If a logic $\mathcal{L}$ has a splitting relation, then the syntax of $\mathcal{S}(\mathcal{L})$ is the extension of $\mathcal{B}(\mathcal{L})$ by the grammar rule $\varphi ::= (\varphi -\!\circ \varphi)$.

Propositional team logic $\mathcal{PTL}$ is the logic of $\mathcal{S}(\mathcal{PL})$-formulas. A valuation of $\mathcal{PTL}$ is a *team* T which is a (possibly empty) set of propositional assignments $s\colon \mathcal{PS} \to \{0, 1\}$. If $\varphi \in \mathcal{PL}$ then $T \vDash \varphi$ if $s \vDash \varphi$ in $\mathcal{PL}$ semantics for all $s \in T$. A division of a team T is simply a pair (S, U) such that $S \cup U = T$.

Modal team logic $\mathcal{MTL}$ is the closure of $\mathcal{ML}$ under ${\sim}, {\to}, -\!\circ$ (as above) and the unary modalities $\Box$ and Δ. Abbreviate $\Diamond := {\sim}\Delta{\sim}$. In contrast to $\mathcal{ML}$, valuations are not pointed Kripke structures $(\mathcal{K}, w)$ but have the form $(\mathcal{K}, T)$, where $T \subseteq W$ is called a *team*. For $\varphi \in \mathcal{ML}$ it holds $(\mathcal{K}, T) \vDash \varphi$ if $(\mathcal{K}, w) \vDash \varphi$ in $\mathcal{ML}$ semantics for all $w \in T$. A division of $(\mathcal{K}, T)$ is a pair $((\mathcal{K}, S), (\mathcal{K}, U))$ such that $S \cup U = T$.

If (W, R, V) is a Kripke structure, then we define the *image* $R[T]$ of a team $T \subseteq W$ as $\{\, w \in W \mid \exists v \in T : vRw \,\}$ and the *pre-image* $R^{-1}[T]$ as $\{\, w \in W \mid \exists v \in T : wRv \,\}$. A *successor team* T' of T is a team such that $T' \subseteq R[T]$ and $T \subseteq R^{-1}[T']$. The semantics of $\Box$ and Δ is $(K, T) \vDash \Box\varphi$ if $(K, R[T]) \vDash \varphi$, and $(K, T) \vDash \Delta\varphi$ if for all successor teams T' of T it holds $(K, T') \vDash \varphi$.

In the following we drop parentheses according to the usual precedence rules; further we assume $\to$, $\twoheadrightarrow$ and $-\!\circ$ as right-associative and $\wedge, \mathbin{⩒}, \vee, \mathbin{⩔}, \otimes$ as left-associative.

Also we reserve the letters $\alpha, \beta, \gamma, \ldots$ for classical $\mathcal{PL}$, $\mathcal{ML}$ formulas; and we use $\varphi, \psi, \vartheta, \ldots$ for general $\mathcal{PTL}$ and $\mathcal{MTL}$ formulas.

2.2 Proof systems

Proof systems or *calculi* are connected to the so-called *Entscheidungsproblem*, the problem of algorithmically deciding if a given formula φ of a logic $\mathcal{L}$ is valid. Formally we define a proof system as a triple $\Omega = (\Xi, \Psi, I)$ where Ξ is a set of formulas, $\Psi \subseteq \Xi$ is a set of *axioms*, and $I \subseteq \mathfrak{P}(\Xi) \times \Xi$ is a set of *inference rules*. Ξ, Ψ and I are all countable and decidable.

An *Ω-proof* $\mathcal{P}$ from a given set of premises $\Phi \subseteq \Xi$ is a finite sequence $\mathcal{P} = (P_1, \ldots, P_n)$ of finite sets $P_i \subseteq \Xi$ such that $\xi \in P_i$ implies $\xi \in P_{i-1} \cup \Psi \cup \Phi$ or $I(P'_{i-1}, \xi)$ for some $P'_{i-1} \subseteq P_{i-1}$. We say that $\mathcal{P}$ *proves* or *derives* a formula φ from Φ if $\varphi \in P_n$ and $\mathcal{P}$ is an Ω-proof from Φ. We write $\Phi \vdash_\Omega \varphi$ if there is some Ω-proof that proves φ from Φ. If Ω is clear then we just write $\Phi \vdash \varphi$. If two formulas φ and φ' prove each other, i.e., $\{\varphi\} \vdash \varphi'$ and $\{\varphi'\} \vdash \varphi$, then we write $\varphi \dashv\vdash \varphi'$. For sets write $\Phi \dashv\vdash \Phi'$ if for every $\varphi \in \Phi$ it holds $\Phi' \vdash \varphi$, and for every $\varphi' \in \Phi'$ it holds $\Phi \vdash \varphi'$.

A calculus Ω is *sound* for a logic $\mathcal{L}$ if for $\Phi \subseteq \mathcal{L}$, $\varphi \in \mathcal{L}$ it holds that $\Phi \vdash_\Omega \varphi$ implies $\Phi \vDash_{\mathcal{L}} \varphi$, and it is *complete* if conversely $\Phi \vDash_{\mathcal{L}} \varphi$ implies $\Phi \vdash_\Omega \varphi$. We say Ω' is *stronger* than Ω, in symbols $\Omega' \succeq \Omega$, if $\Phi \vdash_\Omega \varphi$ implies $\Phi \vdash_{\Omega'} \varphi$. Clearly if Ω' is sound, then Ω is sound, and if Ω is complete, then Ω' is complete. The union of two systems Ω, Ω' is defined as component-wise union and just written $\Omega\Omega'$.

The proof systems presented in this article are based on classical Hilbert-style axiomatizations of propositional and modal logic, as depicted in Figure 1. The propositional system H^0 consists of the axiom schemas (A1)–(A3) and the inference rule (E$\to$) (*modus ponens*). The modal logic K, the weakest normal modal logic, is obtained by the system $\mathsf{H}^\Box$, which

(A1)	$\alpha \to (\beta \to \alpha)$
(A2)	$(\alpha \to (\beta \to \gamma)) \to (\alpha \to \beta) \to (\alpha \to \gamma)$
(A3)	$(\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$
(K)	$\Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$
(E→)	$\dfrac{\alpha \qquad \alpha \to \beta}{\beta}$
(Nec)	$\dfrac{\alpha}{\Box\alpha}$ (α theorem)

Figure 1 Hilbert-style axiomatizations of $\mathcal{PL}$ and $\mathcal{ML}$.

consists of the axiom schemas (A1)–(A3) and the (K) axiom schema as well as the inference rules (E→) and (Nec) (*necessitation*).

In Figure 1, *theorem* means that α was derived without assumptions. Indeed the deduction $\alpha \vdash \Box\alpha$ would not be valid otherwise.

We defined $\mathcal{PL}$ and $\mathcal{ML}$ in team semantics to be *flat*, i.e., to have the *flatness property*: A formula is satisfied by a team T in team semantics exactly when all of T's members satisfy it in classical semantics. In the following we emphasize this by referring to flat logics as $\mathcal{F}$. From the flatness property we can prove that it is unnecessary to distinguish between a classical and a team-semantical entailment relation.

▶ **Proposition 2.1.** *Let $\mathcal{F} \in \{\mathcal{PL}, \mathcal{ML}\}$, $\Gamma \subseteq \mathcal{F}, \alpha \in \mathcal{F}$. Then $\Gamma \models \alpha$ holds in team semantics if and only if it holds in classical semantics.*

Proof. We prove the $\mathcal{PL}$ case. "⇒" follows since assignments are just singleton teams. For "⇐" let $T \models \Gamma$. Then $\forall s \in T : s \models \Gamma$, hence $\forall s \in T : s \models \alpha$. By flatness again $T \models \alpha$. ◀

▶ **Corollary 2.2.** *In team semantics the calculi H^0 and $\mathsf{H}^\Box$ are sound and complete for $\mathcal{PL}$ and $\mathcal{ML}$.*

Other straightforward consequences of flatness are the following, where $\mathcal{F} \in \{\mathcal{PL}, \mathcal{ML}\}$:

▶ **Proposition 2.3** (Downward closure)**.** *If $A \models \alpha$ for a formula $\alpha \in \mathcal{F}$, then $A_1 \models \alpha$ and $A_2 \models \alpha$ for all divisions (A_1, A_2) of A.*

▶ **Proposition 2.4** (Union closure)**.** *If $A_1 \models \alpha$ and $A_2 \models \alpha$ for a formula $\alpha \in \mathcal{F}$, then $A \models \alpha$ for all A which have a division into (A_1, A_2).*

▶ **Proposition 2.5** (Flatness of $\otimes$)**.** *For all $\alpha, \beta \in \mathcal{F}$ it holds $A \models \alpha \vee \beta$ if and only if $A \models \alpha \otimes \beta$.*

Note that propositional and modal team logics are defined in literature using only literals like $p, \neg p$ as atoms. The classical operators $\neg, \vee$ and $\wedge$ (plus $\Box, \Diamond$ in the modal case) are then the primitive connectives (see e.g. Väänänen, Sano and Virtema [13, 14, 15]), where $\vee$ is written $\otimes$ here. With this approach, every team-logical formula with the flatness property is already syntactically identical with the corresponding classical formula.

The rationale behind deviating from this notation is twofold. First, embedding the classical logics with their semantics as its own "layer" in team logic allows to comfortably build onto their proof systems. Second, Hilbert-style proof systems contain introduction rules of the form $\alpha \vdash \Box\alpha$, where α is a tautology. We will show similar introduction rules for all team-logical operators, and such introduction rules for the existential form of the operators, i.e., $\otimes$ and $\Diamond$ instead of $-\!\!\circ$ and Δ, are simply unsound.

(L1)	$\varphi \multimap (\psi \multimap \varphi)$
(L2)	$(\varphi \multimap (\psi \multimap \vartheta)) \multimap (\varphi \multimap \psi) \multimap (\varphi \multimap \vartheta)$
(L3)	$(\sim\varphi \multimap \sim\psi) \multimap (\psi \multimap \varphi)$
(L4)	$(\alpha \to \beta) \multimap (\alpha \multimap \beta)$
(E$\multimap$)	$\dfrac{\varphi \qquad \varphi \multimap \psi}{\psi}$

Figure 2 Hilbert-style axiomatization L of $\mathcal{B}(\mathcal{F})$.

3 Axioms of the Boolean closure

We begin the development of a proof system for team logic with the operators $\multimap$ and $\sim$. They are purely truth-functional; hence we can only reason about Boolean combinations of classical formulas. The presented calculus for this is the system L (for *lifted propositional axioms*) shown in Figure 2. The system L corresponds to the usual propositional axioms, with exception of (L4) which relates the propositional and the material implication. In this section it is shown how any complete proof system for a logic $\mathcal{F}$ can be augmented with L to obtain a complete system for $\mathcal{B}(\mathcal{F})$.

For the team-logical material implication $\multimap$ a new *modus ponens* inference rule is introduced. While the systems H^0 and $\mathsf{H}^\Box$ can only be applied to classical formulas $\alpha, \beta, \gamma, \ldots$, i.e., where no team-logical operators occur, the axioms and rules in L are permitted for general team-logical formulas $\varphi, \psi, \vartheta, \ldots$.

The proof of completeness of L is based on a generalized deduction theorem. The thought behind this strategy is that the deduction theorem implies *Lindenbaum's lemma* which allows the construction of a maximal consistent set, the usual method for completeness proofs of propositional axioms. We begin with identifying a family of proof systems which guarantee a deduction theorem, extending the ideas of Hakli and Negri [5].

▶ **Definition 3.1.** Let $\Omega = (\Xi, \Psi, I)$ be a calculus. Say that a rule $(\{\xi_1, \ldots, \xi_k\}, \psi) \in I$ has *weakening* if $\{\, \varphi \multimap \xi_i \mid i \in [k] \,\} \vdash \varphi \multimap \psi$ for all $\varphi \in \Xi$.

In other words, every derivation using an inference rule can also be proven under arbitrary assumptions. Say that a calculus Ω has weakening if all inference rules have weakening.

▶ **Lemma 3.2.** *If $\Omega \succeq \mathsf{L}$ and Ω has weakening then it has the deduction theorem: $\Phi \vdash (\varphi \multimap \psi)$ if and only if $\Phi \cup \{\varphi\} \vdash \psi$.*

Proof. The direction from left to right is clear as L has (E$\multimap$). From right to left we do an induction over the length n of a shortest proof of ψ. If $\psi \in \Phi$, $\psi = \varphi$, or if ψ is an axiom, then by (L1) and (E$\multimap$) $\Phi \vdash (\varphi \multimap \psi)$. For $n = 1$ these are the only cases. Let $n > 1$. Then ψ could be obtained by application of some inference rule $(\{\xi_1, \ldots, \xi_k\}, \psi)$. $\xi_1, \ldots, \xi_k$ all have a proof of length $\leq n - 1$ from $\Phi \cup \{\varphi\}$, so by induction hypothesis $\Phi \vdash \varphi \multimap \xi_i$ for $i \in [k]$. By weakening $\Phi \vdash \varphi \multimap \psi$. ◀

▶ **Lemma 3.3** (Deduction theorem of L)**.** *If $\Omega \succeq \mathsf{L}$ and all inference rules of Ω except (E$\to$) and (E$\multimap$) yield only theorems (i.e., formulas provable without assumptions), then Ω has the deduction theorem.*

Proof. By the preceding lemma we can instead show that Ω has weakening. If a formula ψ produced by an inference rule is always a theorem, i.e., provable without assumptions,

then by (L1) we can trivially prove $\xi \multimap \psi$ for any ξ. Consider (E$\multimap$). From the assumptions $\xi \multimap (\varphi \multimap \psi)$ and $\xi \multimap \varphi$ just derive $\xi \multimap \psi$ by (L2). For (E$\to$), i.e., $(\{\alpha, \alpha \to \beta\}, \beta)$, weakening is shown as follows: $\theta := \xi \multimap (\alpha \to \beta) \multimap (\alpha \multimap \beta)$ is a theorem due to (L4) and (L1). Apply (L2) twice on the formulas $\xi \multimap (\alpha \to \beta)$, θ and $\xi \multimap \alpha$ to obtain $\xi \multimap \beta$. Hence all rules have weakening. ◀

3.1 Completeness of the Boolean closure

The typical textbook proof of completeness of propositional or first-order logic uses Lindenbaum's lemma to construct a maximal consistent set. For this we need the notion of inconsistency.

▶ **Definition 3.4.** Let $\Omega = (\Xi, \Psi, I)$ be a proof system. A set Φ is *Ω-inconsistent* (or just *inconsistent*) if $\Phi \vdash \Xi$. Φ is *Ω-consistent* (or just *consistent*) if it is not Ω-inconsistent.

▶ **Lemma 3.5.** *Let $\Omega = (\Xi, \Psi, I), \Omega \succeq \mathsf{L}$. The following statements are equivalent:*
1. *$\Phi \vdash \varphi$ and $\Phi \vdash \sim\varphi$ for some φ,*
2. *Φ is inconsistent,*
3. *$\Phi \vdash \perp\!\!\!\perp$.*

Proof. For 1. $\Rightarrow$ 2. we have to show $\Phi \vdash \xi$ for all $\xi \in \Xi$. First $\Phi \vdash (\sim\xi \multimap \sim\varphi)$ follows from $\Phi \vdash \sim\varphi$, (L1) and (E$\multimap$); by (L3) and (E$\multimap$) then follows $\Phi \vdash (\varphi \multimap \xi)$, and again by (E$\multimap$) then $\Phi \vdash \xi$. 3. is a special case of 2. For 3. $\Rightarrow$ 1. it suffices to derive $(\psi \multimap \psi)$ by a textbook proof, since $\perp\!\!\!\perp := \sim(\psi \multimap \psi)$. ◀

▶ **Lemma 3.6** (Relative consistency)**.** *Let $\Omega \succeq \mathsf{L}$ have weakening and let Φ be consistent. Then $\Phi \nvdash \varphi$ implies that $\Phi \cup \{\sim\varphi\}$ is consistent, and $\Phi \vdash \varphi$ implies that $\Phi \cup \{\varphi\}$ is consistent.*

Proof. If $\Phi \nvdash \varphi$ and $\Phi \cup \{\sim\varphi\}$ was inconsistent, then $\Phi \cup \{\sim\varphi\} \vdash \sim\psi$ for any axiom ψ and thus by Lemma 3.2 $\Phi \vdash (\sim\varphi \multimap \sim\psi)$. By (L3) then $\Phi \vdash \psi, \psi \multimap \varphi$, so by (E$\multimap$) $\Phi \vdash \varphi$, contradiction.

If $\Phi \vdash \varphi$ and $\Phi \cup \{\varphi\}$ was inconsistent, then again $\Phi \vdash \varphi, \varphi \multimap \perp\!\!\!\perp$: contradiction to consistency of Φ and Lemma 3.5. ◀

▶ **Definition 3.7.** If $\Omega = (\Xi, \Psi, I)$ then $\Phi \subseteq \Xi$ is *maximal consistent* if it is consistent and contains ξ or $\sim\xi$ for every $\xi \in \Xi$.

▶ **Lemma 3.8** (Lindenbaum's Lemma)**.** *Let $\Omega = (\Xi, \Psi, I), \Omega \succeq \mathsf{L}$. If Ω has weakening, then every consistent set $\Phi \subseteq \Xi$ has a maximal consistent superset $\Phi^* \subseteq \Xi$.*

Proof. Straightforward by enumerating all formulas and applying Lemma 3.6: For every formula ξ, add either ξ or $\sim\xi$. See the appendix for details. ◀

The application of Lindenbaum's lemma is usually as follows: If a set Φ is maximal consistent, then there is a model fulfilling all its atomic formulas. By the maximality of Φ then one can inductively claim that also all Boolean combinations of atomic formulas in Φ are automatically fulfilled as well. The main work here is required for the induction basis — the model satisfying the atomic formulas. In our context, an "atom" is in fact any formula of the underlying classical logic, in this case $\mathcal{PL}$ or $\mathcal{ML}$. This additional complexity requires the next property as an additional step to completeness.

▶ **Definition 3.9.** Let $\mathcal{F}$ be a logic. $\mathcal{F}$ admits *counter-model merging* if it has the following property for arbitrary sets $\Gamma, \Delta \subseteq \mathcal{F}$: If for every $\delta \in \Delta$ there is a valuation falsifying δ and satisfying Γ, then there is a valuation falsifying all formulas in Δ and satisfying Γ.

▶ **Lemma 3.10.** *$\mathcal{PL}$ and $\mathcal{ML}$, under team semantics, admit counter-model merging.*

Proof. We prove only the $\mathcal{ML}$ case as $\mathcal{PL}$ works similar. Let $\Gamma, \Delta \subseteq \mathcal{ML}$. Assume for each $\delta \in \Delta$ a Kripke structure $(\mathcal{K}_\delta, T_\delta)$ that falsifies δ and satisfies Γ. Define $\mathfrak{K}$ as the disjoint union (see Goranko and Otto [3]) of all Kripke structures $\mathcal{K}_\delta$. Then $(\mathfrak{K}, T_\delta) \vDash \Gamma$, $(\mathfrak{K}, T_\delta) \nvDash \delta$ as $\mathcal{ML}$ is invariant under disjoint union of structures [3] and due to flatness of $\mathcal{ML}$. Define the team $\mathcal{T} := \bigcup_{\delta \in \Delta} T_\delta$. As $\mathcal{ML}$ is union closed (Proposition 2.4), $(\mathfrak{K}, \mathcal{T})$ satisfies Γ, and as it is downwards closed (Proposition 2.3), it falsifies each $\delta \in \Delta$. ◀

▶ **Definition 3.11.** A calculus Ω is *refutation complete* for $\mathcal{L}$ if for every unsatisfiable $\Phi \subseteq \mathcal{L}$ there is a φ s. t. $\Phi \vdash \varphi, {\sim}\varphi$.

Write ${\sim}\mathcal{F}$ for the fragment of $\mathcal{B}(\mathcal{F})$ restricted to the formulas $\{ {\sim}\varphi \mid \varphi \in \mathcal{F} \}$.

▶ **Lemma 3.12.** *If $\mathcal{F}$ has counter-model merging and Ω is complete for $\mathcal{F}$, then Ω is refutation complete for $\mathcal{F} \cup {\sim}\mathcal{F}$.*

Proof. Let a set $\Phi \subseteq \mathcal{F} \cup {\sim}\mathcal{F}$ be unsatisfiable. Abbreviate $\Gamma := \Phi \cap \mathcal{F}$ and $\Delta := \Phi \cap {\sim}\mathcal{F}$. It is not the case that $\Gamma \cup \{{\sim}\delta\}$ is satisfiable for every ${\sim}\delta \in \Delta$, because then Φ would be satisfiable by counter-model merging. Hence for some ${\sim}\delta \in \Delta$ the set $\Gamma \cup \{{\sim}\delta\}$ is unsatisfiable, i.e., $\Gamma \vDash \delta$. But then $\Gamma \vdash \delta$ due to the completeness of Ω for $\mathcal{F}$, so $\Phi \vdash \delta, {\sim}\delta$. ◀

Let us emphasize again the difference to classical logics, say, $\mathcal{PL}$: $\mathcal{PL}$ has $\mathcal{PS}$ as its atoms, and the analogously defined fragment $\mathcal{PS} \cup \neg\mathcal{PS}$ of $\mathcal{PL}$ is trivially "refutation complete": A set $\Gamma \subseteq \mathcal{PS} \cup \neg\mathcal{PS}$ is contradictory if and only if contains $p, \neg p$ for some proposition p, so there is nothing to do for a proof system. This is different for team logics.

After the atoms are handled correctly by the proof system (by refutation completeness of $\mathcal{F} \cup {\sim}\mathcal{F}$), the induction step goes just for classical logic, and then results in completeness of $\mathcal{B}(\mathcal{F})$.

▶ **Lemma 3.13.** *If $\Omega \succeq \mathsf{L}$ is refutation complete for $\mathcal{F} \cup {\sim}\mathcal{F}$ and has the deduction theorem, then Ω is refutation complete for $\mathcal{B}(\mathcal{F})$.*

Proof. We must show that every unsatisfiable $\Phi \subseteq \mathcal{B}(\mathcal{F})$ allows deriving φ and ${\sim}\varphi$ for some φ, or, equivalently due to Lemma 3.5, that it is inconsistent. We prove for contraposition that every consistent $\Phi \subseteq \mathcal{B}(\mathcal{F})$ has a model.

If Φ is consistent, then it has a maximal consistent superset Φ^* by Lemma 3.8. Certainly $\Phi^* \cap (\mathcal{F} \cup {\sim}\mathcal{F})$ is consistent as well, and by refutation completeness it has a model $\mathcal{A}$. We show that $\psi \in \Phi^* \Leftrightarrow \mathcal{A} \vDash \psi$ for all $\psi \in \mathcal{B}(\mathcal{F})$ (then Φ^* and in particular Φ is satisfiable).

The rest of the proof will be an induction over the length of ψ. Let $\psi \in \mathcal{F}$. If $\psi \in \Phi^*$, then $\mathcal{A} \vDash \psi$ by definition of $\mathcal{A}$. If $\psi \notin \Phi^*$, then ${\sim}\psi \in \Phi^*$ due to the maximality of Φ^*, so ${\sim}\psi \in \Phi^* \cap {\sim}\mathcal{F}$, and again $\mathcal{A} \vDash {\sim}\psi$ by the definition of $\mathcal{A}$, hence $\mathcal{A} \nvDash \psi$ by definition of ${\sim}$.

The induction step $\psi = {\sim}\vartheta$ is clear due to the consistency and maximality of Φ^*.

So let $\psi = \psi_1 \twoheadrightarrow \psi_2$. Assume $\psi \in \Phi^*$. Then either $\psi_1 \notin \Phi^*$ or ${\sim}\psi_2 \notin \Phi^*$, otherwise by modus ponens Φ^* is inconsistent. But then either $\mathcal{A} \nvDash \psi_1$ or $\mathcal{A} \vDash \psi_2$ by induction hypothesis, hence $\mathcal{A} \vDash \psi_1 \twoheadrightarrow \psi_2$. If $\psi \notin \Phi^*$, then ${\sim}\psi \in \Phi^*$. If now $\mathcal{A} \vDash \psi_2$, then $\psi_2 \in \Phi^*$ by induction hypothesis. From ψ_2 we can derive ψ via (L1). If $\mathcal{A} \vDash {\sim}\psi_1$, then ${\sim}\psi_1 \in \Phi^*$. From ${\sim}\psi_1$ we can infer ${\sim}\psi_2 \twoheadrightarrow {\sim}\psi_1$ again with (L1) and by contraposition (L3) we obtain the conditional ψ. But in both cases Φ would then be inconsistent, so $\mathcal{A} \vDash \psi_1$ and $\mathcal{A} \nvDash \psi_2$, hence $\mathcal{A} \nvDash \psi_1 \twoheadrightarrow \psi_2$. ◀

▶ **Theorem 3.14** (Completeness of L). *If $\Omega \succeq \mathsf{L}$ is refutation complete for $\mathcal{F} \cup {\sim}\mathcal{F}$ and has the deduction theorem, then it is complete for $\mathcal{B}(\mathcal{F})$.*

$(\alpha \otimes \beta) \multimap\!\!\!

▶ **Proposition 4.2.** *There is no finite set $\Phi \subseteq \mathcal{B}(\mathcal{PL})$ that is equivalent to* $\text{NE} \mathbin{⫽} \sim(\text{NE} \otimes \text{NE})$ *in strict semantics.*

Proof. Assume for the sake of contradiction that there was some finite $\Phi \subseteq \mathcal{B}(\mathcal{PL})$ as above. W.l.o.g. the variable x does not occur in Φ. Let $T \vDash \Phi$, then $T = \{s\}$ for some assignment s. It can be easily shown by induction over the length of formulas that $\{s_0^x, s_1^x\} \vDash \Phi$, where $s_c^x(x) = c$ and $s_c^x(y) = s(y)$ for $x \neq y$. ◀

One remark about the naming of the "necessitation" rule in Figure 3. This rule is similar to the rule used in modal logic. In the context of teams, a subteam can as well be seen as a type of "other world". We can, as typical for modal logics, derive no knowledge about ψ in a subteam from knowledge about ψ in the current team. Instead we can see team logics as logics with countable many modalities of the form "$\varphi \multimap$" and introduce corresponding necessitation and distribution rules.

▶ **Proposition 4.3.** *The proof system* $\mathsf{H}^0\mathsf{LS}$ *is sound for* $\mathcal{PTL}$.

Proof. The soundness of H^0 is clear as instances of its axioms may only be $\mathcal{PL}$ formulas and H^0 is sound for $\mathcal{PL}$. The soundness of L is clear as well. So consider the axioms and rules introduced in S: The soundness of $(\mathsf{F}\multimap)$ and $(\mathsf{F}\otimes)$ is due to downward closure (Proposition 2.3) and flatness (Proposition 2.5). (Lax) follows from the definition of splitting, and $(\mathsf{Ex}\multimap)$ and $(\mathsf{C}\multimap)$ can easily be proven by contradiction. The necessitation rule $(\mathsf{Nec}\multimap)$ and the distribution axiom $(\mathsf{Dis}\multimap)$ are as in modal logic, just with the pseudo-modality $\psi \multimap$. $(\mathsf{Nec}\multimap)$ is applied to φ only if φ is a theorem, i.e., $\vdash \varphi$. Its soundness follows therefore straightforwardly by induction over the proof length. ◀

4.1 Completeness of propositional team logic

The proof of Theorem 4.1, the collapse of propositional team logic to the Boolean closure, will be built on several lemmas and the following meta-rules.

▶ **Lemma 4.4.** *If a proof system Ω has the deduction theorem, then it admits the following meta-rules:*

- *If $\Omega \succeq \mathsf{L}$: Reductio ad absurdum (RAA): $\Phi \cup \{\ \varphi\} \vdash \psi, \sim\psi \ \Rightarrow\ \Phi \vdash \sim\varphi$ and $\Phi \cup \{\sim\varphi\} \vdash \psi, \sim\psi \ \Rightarrow\ \Phi \vdash\ \varphi$.*
- *If $\Omega \succeq \mathsf{LS}$: Modus ponens in $\multimap$ (MP$\multimap$): $\vdash \varphi \twoheadrightarrow \psi$, $\Phi \vdash \vartheta \multimap \varphi \ \Rightarrow\ \Phi \vdash \vartheta \multimap \psi$.*
- *If $\Omega \succeq \mathsf{LS}$: Modus ponens in $\otimes$ (MP$\otimes$): $\vdash \varphi \twoheadrightarrow \psi$, $\Phi \vdash \vartheta \otimes \varphi \ \Rightarrow\ \Phi \vdash \vartheta \otimes \psi$.*

Proof. Only a few applications of the deduction theorem and the axioms are required, see the appendix for details. ◀

The next definition is required since our strategy of $\multimap$-elimination starts at the innermost subformulas. Equivalent subformulas can always be substituted in compositional semantics, but we still have to show that $\mathsf{H}^0\mathsf{LS}$ proves these substitutions as well.

▶ **Definition 4.5.** Let $\mathfrak{f}$ be an n-ary connective. Say that a proof system Ω has *substitution in* $\mathfrak{f}$ if $\varphi_i \dashv\vdash \psi_i$ f. a. $i \in [n]$ implies $\mathfrak{f}(\varphi_1, \ldots, \varphi_n) \dashv\vdash \mathfrak{f}(\psi_1, \ldots, \psi_n)$.

Note that due to symmetry it suffices to prove only $\mathfrak{f}(\varphi_1, \ldots, \varphi_n) \vdash \mathfrak{f}(\psi_1, \ldots, \psi_n)$ to show substitution in $\mathfrak{f}$.

▶ **Lemma 4.6.** *If $\Omega \succeq \mathsf{LS}$ has the deduction theorem, then it has substitution in $\sim$, $\twoheadrightarrow$ and $\multimap$.*

$(\varphi \otimes \psi) \leftrightarrow (\psi \otimes \varphi)$	(Com⊗)	Commutative law for $\otimes$
$((\varphi \otimes \psi) \otimes \vartheta) \leftrightarrow (\varphi \otimes (\psi \otimes \vartheta))$	(Ass⊗)	Associative law for $\otimes$
$\alpha \parr (\varphi \otimes \psi) \leftrightarrow (\alpha \parr \varphi) \otimes (\alpha \parr \psi)$	(D⅋⊗)	Distributive law for $\otimes$ and $\parr$
$\varphi \otimes (\psi \vee \vartheta) \leftrightarrow (\varphi \otimes \psi) \vee (\varphi \otimes \vartheta)$	(D∨⊗)	Distributive law for $\otimes$ and $\vee$
$(\varphi \otimes \psi) \parr (\varphi \multimap \vartheta) \twoheadrightarrow (\varphi \otimes (\psi \parr \vartheta))$	(Aug⊗)	
$(\mathsf{E}\alpha \otimes \varphi) \twoheadrightarrow \mathsf{E}\alpha$	(Abs⊗)	Absorption law of $\otimes$
$(\alpha \parr \mathsf{E}\beta) \twoheadrightarrow \mathsf{E}(\alpha \wedge \beta)$	(JoinE)	
$(\varphi \otimes (\alpha \parr \mathsf{E}\beta)) \leftrightarrow (\varphi \otimes \alpha) \parr \mathsf{E}(\alpha \wedge \beta)$	(IsolateE)	

Figure 4 Auxiliary theorems S' of splitting.

Proof. Let $\varphi = \xi_1 \twoheadrightarrow \xi_2$, $\xi_1 \dashv\vdash \psi_1$ and $\xi_2 \dashv\vdash \psi_2$. Then $\{\psi_1, \varphi\} \vdash \xi_2$ in L and hence $\{\psi_1, \varphi\} \vdash \psi_2$. By the deduction theorem $\varphi \vdash \psi_1 \twoheadrightarrow \psi_2$. Let $\varphi = {\sim}\xi$ and $\xi \dashv\vdash \psi$. Obviously $\{\varphi, \psi\} \vdash \xi, {\sim}\xi$ in L. By Lemma 4.4 (RAA) is derivable, so we obtain $\varphi \vdash {\sim}\psi$. The $\multimap$ case is proven with applications of (MP$\multimap$) and (C$\multimap$). ◀

If α is a classical formula, in the following write $\mathsf{E}\alpha$ as an abbreviation for ${\sim}\neg\alpha$. The meaning of $\mathsf{E}\alpha$ is intuitively that at least one element in the current team satisfies α (in particular $\mathsf{E}\alpha$ implies NE).

▶ **Lemma 4.7** ([12])**.** *Let $\Omega \succeq \mathsf{H}^0\mathsf{LS}$ have the deduction theorem. Then Ω proves the theorems S' (see Figure 4).*

We formally describe the translation from $\mathcal{PTL}$ to $\mathcal{B}(\mathcal{PL})$ as *$\multimap$-elimination*. The proof is a step-wise translation with the help of the following lemmas. We implicitly use Lemma 4.6 which permits derivations applied to subformulas inside $\twoheadrightarrow$, ${\sim}$ and $\multimap$, as well as the meta-rules in Lemma 4.4 and the theorems in Lemma 4.7.

▶ **Lemma 4.8.** *If Ω has the deduction theorem and $\Omega \succeq \mathsf{H}^0\mathsf{LS}$, then the following formulas are theorems of Ω:*

$$\bigparr_{i=1}^{n} \mathsf{E}\beta_i \quad \leftrightarrow \bigotimes_{i=1}^{n} \mathsf{E}\beta_i \tag{1}$$

$$\alpha \parr \left(\bigparr_{i=1}^{n} \mathsf{E}\beta_i \right) \leftrightarrow \bigotimes_{i=1}^{n} (\alpha \parr \mathsf{E}\beta_i) \tag{2}$$

$$\bigotimes_{i=1}^{n} (\alpha_i \parr \mathsf{E}\beta_i) \quad \leftrightarrow \left(\bigotimes_{i=1}^{n} \alpha_i \right) \parr \bigparr_{i=1}^{n} \mathsf{E}(\alpha_i \wedge \beta_i) \tag{3}$$

Proof. The proof of $\twoheadrightarrow$ for (1) is by induction over n. $n = 1$ is clear, so let $n > 1$. By induction hypothesis we can assume that $\bigotimes_{i=1}^{n-1} \mathsf{E}\beta_i$ and $\mathsf{E}\beta_n$ are derivable from $\bigparr_{i=1}^{n} \mathsf{E}\beta_i$. For sake of contradiction take $\bigotimes_{i=1}^{n-1} \mathsf{E}\beta_i \multimap {\sim}\mathsf{E}\beta_n$ as assumption; by (Lax) derive $(\top \multimap {\sim}\mathsf{E}\beta_n)$, by (C$\multimap$) then $(\mathsf{E}\beta_n \multimap {\sim}\top)$ and by (Lax) $(\top \multimap {\sim}\top)$. But certainly $\top \vee \top$ is a theorem of $\mathsf{H}^0\mathsf{LS}$ and hence $\top \otimes \top$ due to F$\otimes$. This is a contradiction, so (RAA) yields ${\sim}(\bigotimes_{i=1}^{n} \mathsf{E}\beta_i \multimap {\sim}\mathsf{E}\beta_n)$, i.e., $\bigotimes_{i=1}^{n} \mathsf{E}\beta_i$. The theorems (Abs⊗), (Ass⊗) and (Com⊗) are used to derive each conjunct for $\twoheadleftarrow$ (1).

For (2) first apply (1) to substitute $\bigparr_{i=1}^{n} \mathsf{E}\beta_i$, then distribute α with repeated application of (D⅋⊗), (Ass⊗) and (Com⊗). The reverse direction is possible as both steps are symmetric.

Consider (3). From $\bigotimes_{i=1}^{n}(\alpha_i \parr \mathsf{E}\beta_i)$ we obtain $\bigotimes_{i=1}^{n}\alpha_i$ by (Ass$\otimes$), (Com$\otimes$) and (MP$\otimes$) as $(\alpha_i \parr \mathsf{E}\beta_i) \multimap \alpha_i$ for all $i \in [n]$. Apply (JoinE) to also derive $\bigotimes_{i=1}^{n}\mathsf{E}(\alpha_i \wedge \beta_i)$ the same way, and by (1) then $\bigparr_{i=1}^{n}\mathsf{E}(\alpha_i \wedge \beta_i)$.

For the other implication we repeatedly apply the theorem (IsolateE), i.e., $(\varphi \otimes \alpha) \parr \mathsf{E}(\alpha \wedge \beta) \multimap \varphi \otimes (\alpha \parr \mathsf{E}\beta)$ as follows: Assume that the formula has the following form after k applications.

$$\left(\bigotimes_{i=1}^{k}(\alpha_i \parr \mathsf{E}\beta_i) \otimes \bigotimes_{i=k+1}^{n}\alpha_i\right) \parr \bigparr_{i=k+1}^{n}\mathsf{E}(\alpha_i \wedge \beta_i).$$

For $k = 0$ this is indeed the case. With commutative and associative laws we isolate a single subformula on each side:

$$\left[\left(\bigotimes_{i=1}^{k}(\alpha_i \parr \mathsf{E}\beta_i) \otimes \bigotimes_{i=k+2}^{n}\alpha_i\right) \otimes \alpha_{k+1}\right] \parr \mathsf{E}(\alpha_{k+1} \wedge \beta_{k+1}) \parr \bigparr_{i=k+2}^{n}\mathsf{E}(\alpha_i \wedge \beta_i)$$

Then we apply the theorem on the left two conjuncts, resulting in

$$\left[\left(\bigotimes_{i=1}^{k}(\alpha_i \parr \mathsf{E}\beta_i) \otimes \bigotimes_{i=k+2}^{n}\alpha_i\right) \otimes (\alpha_{k+1} \parr \mathsf{E}\beta_{k+1})\right] \parr \bigparr_{i=k+2}^{n}\mathsf{E}(\alpha_i \wedge \beta_i)$$

and again with commutative and associative laws in

$$\left(\bigotimes_{i=1}^{k+1}(\alpha_i \parr \mathsf{E}\beta_i) \otimes \bigotimes_{i=k+2}^{n}\alpha_i\right) \parr \bigparr_{i=k+2}^{n}\mathsf{E}(\alpha_i \wedge \beta_i)$$

so that we arrive at the same form again and repeat the steps. ◀

▶ **Lemma 4.9** (Flatness Properties). *If Ω has the deduction theorem and $\Omega \succeq \mathsf{H}^0\mathsf{LS}$, then the following formulas are theorems of Ω:*

$$\bigotimes_{i=1}^{n}\alpha_i \leftrightarrow \bigvee_{i=1}^{n}\alpha_i \qquad (4) \qquad\qquad \bigparr_{i=1}^{n}\alpha_i \leftrightarrow \bigwedge_{i=1}^{n}\alpha_i \qquad (5)$$

Proof. (4): By induction over n. The case $n = 1$ is trivial. For $n > 1$ let $\varphi := \bigotimes_{i=1}^{n-1}\alpha_i$ and $\gamma := \bigvee_{i=1}^{n-1}\alpha_i$ be given as assumptions. By induction hypothesis $\varphi \dashv\vdash \gamma$. Then $\varphi \otimes \alpha_n \dashv\vdash \gamma \otimes \alpha_n$ by (Com$\otimes$) and (MP$\otimes$), and by (F$\otimes$) we obtain $\varphi \otimes \alpha_n \dashv\vdash \gamma \otimes \alpha_n \dashv\vdash \gamma \vee \alpha_n$ and hence (4).

(5): Again by induction, so let $\varphi := \bigparr_{i=1}^{n-1}\alpha_i$ and $\gamma := \bigwedge_{i=1}^{n-1}\alpha_i$ be given such that $\varphi \dashv\vdash \gamma$. Then $\varphi \parr \alpha_n \dashv\vdash \gamma \parr \alpha_n$ in L. To obtain $\varphi \parr \alpha_n \dashv\vdash \gamma \parr \alpha_n \dashv\vdash \gamma \wedge \alpha_n$ we prove the general theorem $\alpha \wedge \beta \dashv\vdash \alpha \parr \beta$ for classical α, β. Clearly $\alpha \wedge \beta$ proves α and β in H^0 and thus $\alpha \parr \beta$ in $\mathsf{H}^0\mathsf{L}$. To prove $\alpha \wedge \beta$ from $\alpha \parr \beta$ we use (RAA), i.e., assume $\alpha \parr \beta$ and $\sim(\alpha \wedge \beta)$. It holds $(\alpha \wedge \beta) \dashv\vdash \neg(\alpha \to \neg\beta)$ in H^0, so we derive $\mathsf{E}(\alpha \to \neg\beta)$ from $\sim(\alpha \wedge \beta)$. Via L and the theorem (JoinE) we obtain in two steps first $\mathsf{E}(\alpha \wedge (\alpha \to \neg\beta))$ and then $\mathsf{E}(\beta \wedge \alpha \wedge (\alpha \to \neg\beta))$ which turns into $\mathsf{E}\bot$ in $\mathsf{H}^0\mathsf{L}$. But $\mathsf{E}\bot = \sim\neg\bot$ proves $\perp\!\!\!\perp$, contradiction. ◀

With the above lemmas we are ready to state the full translation from $\mathcal{PTL}$ to $\mathcal{B}(\mathcal{PL})$.

▶ **Definition 4.10.** Let $\mathcal{F}$ be a logic. Let $\Omega = (\Xi, \Psi, I)$ be a proof system. Let $\mathfrak{f}$ be a connective of arity n. We say that $\mathcal{B}(\mathcal{F})$ has $\mathfrak{f}$-*elimination in* Ω if the following holds for all formulas $\xi_1, \ldots, \xi_n \in \Xi$: If for all $i \in [n]$ there is $\xi_i' \in \mathcal{B}(\mathcal{F})$ s. t. $\xi_i \dashv\vdash \xi_i'$, then also $\mathfrak{f}(\xi_1, \ldots, \xi_n) \dashv\vdash \varphi$ for some $\varphi \in \mathcal{B}(\mathcal{F})$.

In other words: If $\xi_1, \dots, \xi_n$ have provably equivalent $\mathcal{B}(\mathcal{F})$ formulas, then so has $\mathfrak{f}(\xi_1, \dots, \xi_n)$.

▶ **Lemma 4.11** (—∘-elimination). *Let $\mathcal{F}$ be a logic closed under $\neg, \vee, \wedge$. Let Ω be a proof system with the deduction theorem s. t. $\Omega \succeq \mathsf{H}^0\mathsf{LS}$. Then $\mathcal{B}(\mathcal{F})$ has —∘-elimination in Ω.*

Proof. Let $\varphi = \psi \multimap \vartheta$. Let $\psi', \vartheta' \in \mathcal{B}(\mathcal{F})$ such that $\psi \dashv\vdash \psi'$ and $\vartheta \dashv\vdash \vartheta'$. It holds $\vartheta' \dashv\vdash \sim\sim\vartheta'$ in L. Lemma 4.6 applies to Ω, so by substitution in $\multimap$ we can translate $\varphi := \psi \multimap \vartheta$ to $\psi' \multimap \sim\sim\vartheta'$, and therefore in L to $\sim\sim(\psi' \multimap \sim\sim\vartheta') = \sim(\psi' \otimes \sim\vartheta')$.

In the system L we can apply De Morgan's laws and distributive laws on both ψ' and $\sim\vartheta'$. We can therefore derive two formulas ψ'', ϑ'' in disjunctive normal form (DNF) over $⅋, \Cap$; and obtain with substitution in $\sim$ and $\multimap$ the following equivalent form of φ, where all $\alpha, \beta, \gamma, \delta, \ldots \in \mathcal{F}$:

$$\sim\left[\bigvee_{i=1}^{n}\left(\bigparr_{j=1}^{o_i}\alpha_{i,j} ⅋ \bigparr_{j=1}^{m_i}\mathsf{E}\beta_{i,j}\right) \otimes \bigvee_{i=1}^{n'}\left(\bigparr_{j=1}^{o'_i}\alpha'_{i,j} ⅋ \bigparr_{j=1}^{m'_i}\mathsf{E}\beta'_{i,j}\right)\right]$$

That the negative literals can be represented with E prefix is due to H^0 proving $\neg\neg$ introduction, thus $\sim\beta \dashv\vdash \sim\neg\neg\beta = \mathsf{E}\neg\beta$ for any β. Apply the following derivation in $\mathsf{H}^0\mathsf{LS}$:

$$\begin{aligned}
&\text{Lemma 4.9 (5)} \quad \dashv\vdash \sim\left[\bigvee_{i=1}^{n}\left(\alpha_i ⅋ \bigparr_{j=1}^{m_i}\mathsf{E}\beta_{i,j}\right) \otimes \bigvee_{i=1}^{n'}\left(\alpha'_i ⅋ \bigparr_{j=1}^{m'_i}\mathsf{E}\beta'_{i,j}\right)\right] \quad (\alpha_i, \alpha'_i \in \mathcal{F})\\
&\substack{(\mathsf{D}⅋\otimes)\\(\mathsf{Com}\otimes)\\(\mathsf{Ass}\otimes)} \quad \dashv\vdash \sim\bigvee_{\substack{1\le i\le n\\1\le i'\le n'}}\left(\alpha_i ⅋ \bigparr_{j=1}^{m_i}\mathsf{E}\beta_{i,j}\right) \otimes \left(\alpha'_{i'} ⅋ \bigparr_{j=1}^{m'_{i'}}\mathsf{E}\beta'_{i',j}\right)\\
&\text{Lemma 4.8 (2)} \quad \dashv\vdash \sim\bigvee_{\substack{1\le i\le n\\1\le i'\le n'}}\left(\bigotimes_{j=1}^{m_i}\left(\alpha_i ⅋ \mathsf{E}\beta_{i,j}\right) \otimes \bigotimes_{j=1}^{m'_{i'}}\left(\alpha'_{i'} ⅋ \mathsf{E}\beta'_{i',j}\right)\right)\\
&\text{(Renaming)} \quad \dashv\vdash \sim\bigvee_{i=1}^{\ell}\bigotimes_{j=1}^{k_i}\left(\gamma_{i,j} ⅋ \mathsf{E}\delta_{i,j}\right)\\
&\text{Lemma 4.8 (3)} \quad \dashv\vdash \sim\bigvee_{i=1}^{\ell}\left(\bigotimes_{j=1}^{k_i}\gamma_{i,j} ⅋ \bigparr_{j=1}^{k_i}\mathsf{E}\left(\gamma_{i,j}\wedge\delta_{i,j}\right)\right)\\
&\text{Lemma 4.9 (4)} \quad \dashv\vdash \sim\bigvee_{i=1}^{\ell}\left(\bigvee_{j=1}^{k_i}\gamma_{i,j} ⅋ \bigparr_{j=1}^{k_i}\mathsf{E}\left(\gamma_{i,j}\wedge\delta_{i,j}\right)\right) \quad =: \varphi' \in \mathcal{B}(\mathcal{F}).
\end{aligned}$$

◀

▶ **Theorem 4.12.** *Every $\mathcal{PTL}$ formula is provably equivalent to a $\mathcal{B}(\mathcal{PL})$ formula in $\mathsf{H}^0\mathsf{LS}$.*

Proof. Let $\varphi \in \mathcal{PTL}$. We show derivability of an equivalent $\varphi' \in \mathcal{B}(\mathcal{PL})$ by induction over $|\varphi|$. So assume $\varphi \notin \mathcal{B}(\mathcal{PL})$. If $\varphi = \psi \twoheadrightarrow \vartheta$ or $\varphi = \sim\psi$ then we have only to apply the induction hypothesis to ψ and ϑ and substitute in the sense of Lemma 4.6. The remaining case is $\varphi = \psi \multimap \vartheta$ for which we apply the previous lemma. ◀

▶ **Lemma 4.13.** *Let $\mathcal{F}, \mathcal{L}$ be logics, $\mathcal{B}(\mathcal{F}) \subseteq \mathcal{L}$, s. t. every $\mathcal{L}$ formula is provably equivalent to a $\mathcal{B}(\mathcal{F})$ formula in Ω. If Ω is complete for $\mathcal{B}(\mathcal{F})$ and sound for $\mathcal{L}$, then Ω is complete for $\mathcal{L}$.*

Proof. Assume $\Phi \subseteq \mathcal{L}, \varphi \in \mathcal{L}$. For completeness we have to show that $\Phi \vDash \varphi$ implies $\Phi \vdash \varphi$ in Ω. By assumption every $\mathcal{L}$ formula is provably equivalent to a $\mathcal{B}(\mathcal{F})$ formula, so $\Phi \dashv\vdash \Phi'$ for some set $\Phi' \subseteq \mathcal{B}(\mathcal{F})$. Similar $\varphi \dashv\vdash \varphi'$ for some $\varphi' \in \mathcal{B}(\mathcal{F})$.

By soundness it holds $\Phi \equiv \Phi'$ and $\varphi \equiv \varphi'$, so $\Phi' \models \varphi'$ follows. By completeness of Ω for $\mathcal{B}(\mathcal{F})$ we have $\Phi' \vdash \varphi'$, and so $\Phi \vdash \Phi' \vdash \varphi' \vdash \varphi$. The lemma follows as $\vdash$ is transitive. ◀

▶ **Theorem 4.14.** *The proof system* $\mathsf{H}^0\mathsf{LS}$ *is sound and complete for* $\mathcal{PTL}$.

Proof. For soundness see Proposition 4.3. $\mathsf{H}^0\mathsf{L}$ is complete for $\mathcal{B}(\mathcal{PL})$ due to Corollary 3.15. By soundness, Theorem 4.12, and Lemma 4.13 we obtain completeness for $\mathcal{PTL}$. ◀

It follows that axiomatizations of $\mathcal{PTL}$-definable constraints, like dependence, independence and inclusion over Boolean relations, can automatically be found in the presented proof system. For instance the dependency atom $=(x, y)$ ("y is a function of x") can be written as $\top \multimap (=(x) \twoheadrightarrow =(y))$, where $=(p) := p \otimes \neg p$.

▶ **Example 4.15.** We give a proof of Armstrong's axiom of transitivity of functional dependence [1]. The axiom says the following: From $=(x, y)$ and $=(y, z)$ infer $=(x, z)$. A proof sketch follows.

A	$=(x, y)$	
B	$=(y, z)$	
1	$(=(x) \twoheadrightarrow =(y)) \twoheadrightarrow ((=(y) \twoheadrightarrow =(z)) \twoheadrightarrow (=(x) \twoheadrightarrow =(z)))$	L
2	$\top \multimap \big((=(x) \twoheadrightarrow =(y)) \twoheadrightarrow ((=(y) \twoheadrightarrow =(z)) \twoheadrightarrow (=(x) \twoheadrightarrow =(z)))\big)$	Nec$\multimap$ (1)
3	$\big(\top \multimap (=(x) \twoheadrightarrow =(y))\big) \twoheadrightarrow \big(\top \multimap ((=(y) \twoheadrightarrow =(z)) \twoheadrightarrow (=(x) \twoheadrightarrow =(z)))\big)$	Dis$\multimap$ + E$\twoheadrightarrow$ (2)
4	$\top \multimap ((=(y) \twoheadrightarrow =(z)) \twoheadrightarrow (=(x) \twoheadrightarrow =(z)))$	E$\twoheadrightarrow$ (A + 3)
5	$\big(\top \multimap (=(y) \twoheadrightarrow =(z))\big) \twoheadrightarrow \big(\top \multimap (=(x) \twoheadrightarrow =(z))\big)$	Dis$\multimap$ + E$\twoheadrightarrow$ (4)
6	$\top \multimap (=(x) \twoheadrightarrow =(z))$	E$\twoheadrightarrow$ (B + 5)
	$=(x, z)$	

▶ **Example 4.16.** The formula $(\alpha \multimap \beta) \twoheadrightarrow \beta$ is valid for all $\alpha, \beta \in \mathcal{PL}$: α is satisfied by the empty team, and as for every team T there is a division into $\emptyset \cup T$, the team T should satisfy β. We sketch a proof in the system $\mathsf{H}^0\mathsf{LS}$: It holds $\mathsf{H}^0 \vdash \bot \to \alpha$, thus $\mathsf{H}^0\mathsf{LS} \vdash {\sim}\alpha \twoheadrightarrow {\sim}\bot$. From ${\sim}\alpha \twoheadrightarrow {\sim}\bot$ and $\alpha \multimap \beta$ it follows $\bot \multimap \beta$ by (C$\multimap$) and (MP$\multimap$), hence ${\sim}\beta \multimap {\sim}\bot$. To prove β now we assume ${\sim}\beta$ for (RAA). From ${\sim}\beta$ and ${\sim}\beta \multimap {\sim}\bot$ we obtain $\top \multimap {\sim}\bot$ by (Lax) which contradicts $\top \otimes \bot := {\sim}(\top \multimap {\sim}\bot)$, but $\top \otimes \bot$ follows from $\mathsf{H}^0 \vdash \top \vee \bot$ and (F$\otimes$).

5 Modal team logic

In modal team logic we have team-wide modalities to relate teams of worlds in Kripke structures to each other. As for the splitting operator, we axiomatize the modalities just enough so that we can eliminate them. Kontinen, Müller, Schnoor and Vollmer [9] proved that every $\mathcal{MTL}$ formula is equivalent to an $\mathcal{B}(\mathcal{ML})$ formula. But as with Yang's argument we improve this result by giving a purely syntactical derivation procedure which does not rely on model-theoretic aspects. Together with Corollary 3.15 this yields a complete axiomatization for $\mathcal{MTL}$, settling an open question of Kontinen et al. [9].

▶ **Proposition 5.1.** *The proof system* $\mathsf{H}^\Box\mathsf{LSM}$ *is sound for* $\mathcal{MTL}$.

Proof. For $\mathsf{H}^\Box$ see Corollary 2.2 and for LS see the proof of Proposition 4.3. The axioms and rules of M can be verified from the definition of $\mathcal{MTL}$, their soundness is shown in the appendix. ◀

$\Box\sim\varphi \leftrightarrow \sim\Box\varphi$	(Lin$\Box$)	$\Box$ is linear.
$\Diamond\alpha \leftrightarrow \neg\Box\neg\alpha$	(F$\Diamond$)	Flatness of $\Diamond$.
$\Diamond(\varphi \otimes \psi) \leftrightarrow \Diamond\varphi \otimes \Diamond\psi$	(D$\Diamond\otimes$)	$\Diamond$ distributes over splitting.
$\Box\alpha \mathrel{-\!\!\triangleright} \Delta\alpha$	(E$\Box$)	Successors are subteams of image team.
$\Diamond\varphi \mathrel{-\!\!\triangleright} (\Delta\psi \mathrel{-\!\!\triangleright} \Box\psi)$	(I$\Box$)	Image is a successor if one exists.
$\Box(\varphi \mathrel{-\!\!\triangleright} \psi) \mathrel{-\!\!\triangleright} (\Box\varphi \mathrel{-\!\!\triangleright} \Box\psi)$	(Dis$\Box$)	Distribution axiom
$\Delta(\varphi \mathrel{-\!\!\triangleright} \psi) \mathrel{-\!\!\triangleright} (\Delta\varphi \mathrel{-\!\!\triangleright} \Delta\psi)$	(DisΔ)	Distribution axiom
$\dfrac{\varphi}{\Box\varphi}$ (φ theorem)	(Nec$\Box$)	Necessitation
$\dfrac{\varphi}{\Delta\varphi}$ (φ theorem)	(NecΔ)	Necessitation

Figure 5 The modal team logic axioms M.

$\Box(\varphi \mathrel{-\!\!\triangleright} \psi) \leftrightarrow (\Box\varphi \mathrel{-\!\!\triangleright} \Box\psi)$	(D$\Box\mathrel{-\!\!\triangleright}$)	Distributive law for $\Box$ and $\mathrel{-\!\!\triangleright}$
$\Diamond(\varphi \mathbin{⫽} \psi) \leftrightarrow (\Diamond\varphi \mathbin{⫽} \Diamond\psi)$	(D$\Diamond⫽$)	Distributive law for $\Diamond$ and $⫽$
$\Diamond(\alpha \mathbin{⫽} \mathsf{E}\beta) \leftrightarrow \Diamond\alpha \mathbin{⫽} \mathsf{E}\neg\Box\neg(\alpha \wedge \beta)$	($\Diamond$IsolateE)	

Figure 6 Auxiliary theorems M′ for modalities.

▶ **Lemma 5.2.** H$^{\Box}$LSM *has substitution in* $\mathrel{-\!\!\triangleright}, \sim, \mathrel{-\!\!\circ}, \Box$ *and* Δ.

Proof. H$^{\Box}$LSM has the deduction theorem due to Lemma 3.3, so the first three cases follow from Lemma 4.6. The cases $\varphi = \Box\xi$ and $\varphi = \Delta\xi$ are easily shown with (Nec$\Box$), (Dis$\Box$), (NecΔ) and (DisΔ). ◀

▶ **Lemma 5.3** ([12])**.** H$^{\Box}$LSM *proves the theorems* M′ *(see Figure 6).*

▶ **Lemma 5.4.** $\mathcal{B}(\mathcal{ML})$ *has* $\Box$*-elimination in* H$^{\Box}$LM.

Proof. Transform the argument of $\Box$ applying Lemma 5.2, the rest follows immediately from the axiom (Lin$\Box$) as well as the distributive law (D$\Box\mathrel{-\!\!\triangleright}$): Just push the $\Box$ inwards until it precedes only classical $\mathcal{ML}$ subformulas. ◀

▶ **Lemma 5.5.** $\mathcal{B}(\mathcal{ML})$ *has* Δ*-elimination in* H$^{\Box}$LSM.

Proof. Let $\Delta\varphi$ be given s. t. $\varphi \dashv\vdash \varphi'$ for $\varphi' \in \mathcal{B}(\mathcal{ML})$. With L and Lemma 5.2 we can prove $\Delta\varphi$ equivalent to $\sim\sim\Delta\sim\sim\varphi' = \sim\Diamond\sim\varphi'$. Again in L we can apply De Morgan's laws and distributive laws on $\sim\varphi'$ such that it is provably equivalent to a formula in DNF:

$$\bigotimes_{i=1}^{n} \left(\bigotimes_{j=1}^{o_i} \alpha_{i,j} \mathbin{⫽} \bigotimes_{j=1}^{k_i} \mathsf{E}\beta_{i,j} \right)$$

Then $\Delta\varphi$ itself is provably equivalent to:

$$\sim \Diamond \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{o_i} \alpha_{i,j} \oslash \bigwedge_{j=1}^{k_i} \mathsf{E}\beta_{i,j} \right)$$

Lemma 4.8, 4.9 $\dashv\vdash$ $\sim \Diamond \bigvee_{i=1}^{n} \bigotimes_{j=1}^{k_i} \left(\alpha_i \oslash \mathsf{E}\beta_{i,j} \right)$ where $\alpha_i \in \mathcal{ML}$

(D$\Diamond\otimes$), (D$\Diamond\otimes$) $\dashv\vdash$ $\sim \bigvee_{i=1}^{n} \bigotimes_{j=1}^{k_i} \Diamond (\alpha_i \oslash \mathsf{E}\beta_{i,j})$

Lemma 5.3 $\dashv\vdash$ $\sim \bigvee_{i=1}^{n} \bigotimes_{j=1}^{k_i} (\Diamond\alpha_i \oslash \mathsf{E}\neg\Box\neg(\alpha_i \wedge \beta_{i,j}))$

(F$\Diamond$) $\dashv\vdash$ $\sim \bigvee_{i=1}^{n} \bigotimes_{j=1}^{k_i} (\neg\Box\neg\alpha_i \oslash \mathsf{E}\neg\Box\neg(\alpha_i \wedge \beta_{i,j}))$

(Renaming) $\dashv\vdash$ $\sim \bigvee_{i=1}^{n} \bigotimes_{j=1}^{k_i} (\mu_{i,j} \oslash \mathsf{E}\nu_{i,j})$ where $\mu_{i,j}, \nu_{i,j} \in \mathcal{ML}$

Lemma 4.8, 4.9 $\dashv\vdash$ $\sim \bigvee_{i=1}^{\ell} \left(\bigvee_{j=1}^{k_i} \mu_{i,j} \oslash \bigwedge_{j=1}^{k_i} \mathsf{E}\left(\mu_{i,j} \wedge \nu_{i,j}\right) \right) \in \mathcal{B}(\mathcal{ML}).$ ◀

▶ **Theorem 5.6.** *Every $\mathcal{MTL}$ formula is provably equivalent to a $\mathcal{B}(\mathcal{ML})$ formula in* $\mathsf{H}^{\Box}\mathsf{LSM}$.

Proof. Proven by induction as in Theorem 4.12, applying Lemma 4.11, 5.4 and 5.5. ◀

▶ **Theorem 5.7.** *The proof system* $\mathsf{H}^{\Box}\mathsf{LSM}$ *is sound and complete for $\mathcal{MTL}$.*

Proof. For the soundness see Proposition 5.1. The completeness follows from soundness, Corollary 3.15, Lemma 4.13 and Theorem 5.6. ◀

6 Conclusion

The team-semantical extensions of propositional logic $\mathcal{PL}$ and modal logic $\mathcal{ML}$, i.e., $\mathcal{PTL}$ and $\mathcal{MTL}$, have been shown axiomatizable. Their property to collapse to the Boolean closures of their flat base logics, i.e., $\mathcal{B}(\mathcal{PL})$ and $\mathcal{B}(\mathcal{ML})$, allows a completeness proof for the given proof systems.

An important detail there is the use of lax semantics for the operators $\otimes$ and $\Diamond$. It is possible to use strict semantics, i.e., to define team divisions via partitions; and to choose exactly one successor of worlds for $\Diamond$ (see Hella et al. [6]). The semantics of $\otimes$ would then allow to *count* certain elements in the team. The strictness of $\Diamond$ can be chosen accordingly: It distributes over $\otimes$ if and only if both are strict or both are lax.

But even if we recover the distributive laws at this point — counting cannot be expressed in the $\mathcal{B}(\cdot)$ closure (see Proposition 4.2), so there can be no completeness proof based on full operator elimination as in the style of this paper. It is open how complete axiomatizations can be found for team logics strictly stronger than $\mathcal{B}(\cdot)$.

Acknowledgements. I thank Anselm Haak and Juha Kontinen for various comments, hints and for pointing out helpful references. I also thank the anonymous referees for their useful hints and corrections.

References

1. William Ward Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.
2. 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.
3. Valentin Goranko and Martin Otto. Model theory of modal logic. In Johan Van Benthem Patrick Blackburn and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 249–329. Elsevier, 2007. doi:10.1016/S1570-2464(07)80008-5.
4. Erich Grädel and Jouko Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.
5. Raul Hakli and Sara Negri. Does the deduction theorem fail for modal logic? *Synthese*, 187(3):849–867, 2012. doi:10.1007/s11229-011-9905-9.
6. Lauri Hella, Antti Kuusisto, Arne Meier, and Heribert Vollmer. Modal Inclusion Logic: Being Lax is Simpler than Being Strict. In Giuseppe F Italiano, Giovanni Pighizzini, and Donald T. Sannella, editors, *Mathematical Foundations of Computer Science 2015*, volume 9234, pages 281–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
7. 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. doi:10.1016/S0049-237X(08)70066-1.
8. Wilfrid Hodges. Compositional semantics for a language of imperfect information. *Logic Journal of the IGPL*, 5(4):539–563, 1997.
9. 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.
10. Juha Kontinen and Ville Nurmi. Team Logic and Second-Order Logic. In *Logic, Language, Information and Computation*, volume 5514, pages 230–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
11. Juha Kontinen and Jouko Väänänen. Axiomatizing first-order consequences in dependence logic. *Annals of Pure and Applied Logic*, 164(11):1101–1117, November 2013. doi:10.1016/j.apal.2013.05.006.
12. Martin Lück. The axioms of team logic. *CoRR*, abs/1510.08786, 2016. URL: http://arxiv.org/abs/1510.08786.
13. Katsuhiko Sano and Jonni Virtema. Axiomatizing Propositional Dependence Logics. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, volume 41 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 292–307, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2015.292.
14. Jouko Väänänen. Modal dependence logic. *New perspectives on games and interaction*, 4:237–254, 2008.
15. 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.
16. Fan Yang. *On extensions and variants of dependence logic*. PhD thesis, University of Helsinki, 2014. URL: http://www.math.helsinki.fi/logic/people/fan.yang/dissertation_fyang.pdf.

A Appendix

Proof of Lemma 3.8 (Lindenbaum's Lemma). Let Φ be consistent, $\Omega = (\Xi, \Psi, I)$. We can write the countable set Ξ as $\Xi := \{\xi_1, \xi_2, \ldots\}$.

Let $\Phi_0 := \Phi$, and for each $i \geq 1$ define Φ_i as

$$\Phi_i := \begin{cases} \Phi_{i-1} \cup \{\xi_i\} & \text{if } \Phi_{i-1} \vdash \{\xi_i\}, \\ \Phi_{i-1} \cup \{\sim\xi_i\} & \text{otherwise.} \end{cases}$$

By Lemma 3.6 the consistency of Φ_{i-1} implies that of Φ_i. Hence by induction all Φ_n for $n \geq 0$ are consistent. Let $\Phi^* := \bigcup_{n\geq 0} \Phi_n$. Φ^* is again consistent, otherwise it could prove $\perp\!\!\!\perp$ already from a finite set Φ_n of assumptions, which would be a contradiction to the consistency of all Φ_n. Φ^* is maximal by construction. ◀

Proof of Lemma 4.4. In the first case of (RAA) we have $\Phi \vdash \varphi \twoheadrightarrow \sim\psi, \varphi \twoheadrightarrow \psi$ by the deduction theorem. L proves the Boolean tautologies $(\varphi \twoheadrightarrow \sim\psi) \twoheadrightarrow (\psi \twoheadrightarrow \sim\varphi)$ and $(\varphi \twoheadrightarrow \sim\varphi) \twoheadrightarrow \sim\varphi$, hence $\Phi \vdash \sim\varphi$ by (E$\twoheadrightarrow$). The second case is proven with the theorem $\sim\sim\varphi \twoheadrightarrow \varphi$ of L. (MP$\multimap$) is just a shortcut for (Nec$\multimap$), (Dis$\multimap$) and (E$\twoheadrightarrow$). (MP$\otimes$) is proven as follows.

A	$\vdash \varphi \twoheadrightarrow \psi$	MP$\otimes$
B	$\{\vartheta \otimes \varphi\}$	
1	$\vdash \sim\psi \twoheadrightarrow \sim\varphi$	L (A)
2	$\vdash (\vartheta \multimap \sim\psi) \twoheadrightarrow (\vartheta \multimap \sim\varphi)$	Nec$\multimap$, Dis$\multimap$ (1)
3	$\sim(\vartheta \multimap \sim\varphi)$	Def. (B)
4	$\sim(\vartheta \multimap \sim\psi)$	L (2 + 3)
5	$\vartheta \otimes \psi$	Def. (4)
	$\vartheta \otimes \psi$	

◀

Proof of Proposition 5.1 (Soundness of H$^\Box$LSM). The system H$^\Box$ applies only to $\mathcal{ML}$ formulas and is hence sound by Corollary 2.2. The system L is easily confirmed sound, and the soundness of S is proven as in Proposition 4.3. So we prove only the axioms M.

(Lin$\Box$): Assume $(\mathcal{K}, T) \vDash \Box\sim\varphi$. Then the unique successor team $R[T]$ does not satisfy φ. So it is not the case that $(\mathcal{K}, R[T]) \vDash \varphi$, hence $(\mathcal{K}, T) \nvDash \Box\varphi$ by definition of $\Box$. The other direction is similar. The flatness axiom (F$\Diamond$) follows from the definition of a successor team. (E$\Box$), (I$\Box$), (Dis$\Box$), (DisΔ) are clear, and as well are (Nec$\Box$) and (NecΔ): If a formula φ is a theorem and hence holds in all teams, then it certainly holds for the image team of any team and all successor teams in general.

It remains to prove (D$\Diamond\otimes$).

"$\twoheadrightarrow$": Assume $\mathcal{K} = (W, R, V)$ and $(\mathcal{K}, T) \vDash \Diamond(\varphi \otimes \psi)$. Then T has a successor team T' which satisfies $\varphi \otimes \psi$, i.e., there are S' and U' such that $T' = S' \cup U'$, $(\mathcal{K}, S') \vDash \varphi$ and $(\mathcal{K}, U') \vDash \psi$. We have to find a split (S, U) of T such that $(\mathcal{K}, S) \vDash \Diamond\varphi$ and $(\mathcal{K}, U) \vDash \Diamond\psi$. Define $S := \{\, v \in T \mid vRu, u \in S' \,\}$ and $U := \{\, v \in T \mid vRu, u \in U' \,\}$. Now every world $v \in T$ has at least one successor $u \in T'$ as T' is a successor team of T. Since $S' \cup U' = T'$, it is either $u \in S'$ or $u \in U'$. Hence by definition of S and U, v is in S or U. So $T \subseteq S \cup U$, and by definition of S and U then $T = S \cup U$. For proving $(\mathcal{K}, S) \vDash \Diamond\varphi$ and $(\mathcal{K}, U) \vDash \Diamond\psi$ it remains to show that S' is really a successor team of S (the proof for U is similar). Certainly every $v \in S$ must have at least one successor in S' by definition of S. Also every $u \in S'$ has

at least one predecessor in S: As $S' \subseteq T'$ and T' is a successor team of T, it holds that u has a predecessor in T, say, v, but then $v \in T$, vRu and $u \in S'$, so $v \in S$ by definition of S. So S' is a successor team of S.

"$\Leftarrow$": Assume $(\mathcal{K}, T) \vDash \Diamond\varphi \otimes \Diamond\psi$ witnessed by $T = S \cup U$, $(\mathcal{K}, S') \vDash \varphi$ and $(\mathcal{K}, U') \vDash \psi$, S', U' being successor teams of S and U. Then $T' = S' \cup U'$ is a successor team of T which witnesses $(\mathcal{K}, T) \vDash \Diamond(\varphi \otimes \psi)$. $(\mathcal{K}, S' \cup U') \vDash \varphi \otimes \psi$ is clear, so we prove that T' is an actual successor team of T. If $v \in T$ then $v \in S$ or $v \in U$, so v has a successor u in S' or U', but either way in T'. If $u \in T'$ then $u \in S'$ or $u \in U'$, so u has a predecessor v in S or U and hence in T. ◀

Semantics for "Enough-Certainty" and Fitting's Embedding of Classical Logic in S4

Gergei Bana[*1] **and Mitsuhiro Okada**[†2]

1 **INRIA de Paris, Paris, France**
bana@math.upenn.edu
2 **Department of Philosophy, Keio University, Tokyo, Japan**
mitsu@abelard.flet.keio.ac.jp

Abstract

In this work we look at how Fitting's embedding of first-order classical logic into first-order S4 can help in reasoning when we are interested in satisfaction "in most cases", when first-order properties are allowed to fail in cases that are considered insignificant. We extend classical semantics by combining a Kripke-style model construction of "significant" events as possible worlds with the forcing-Fitting-style semantics construction by embedding classical logic into S4. We provide various examples. Our main running example is an application to symbolic security protocol verification with complexity-theoretic guarantees. In particular, we show how Fitting's embedding emerges entirely naturally when verifying trace properties in computer security.

1998 ACM Subject Classification F.3.1 Logics and Meanings of Programs

Keywords and phrases first-order logic, possible-world semantics, Fitting embedding, asymptotic probabilities, verification of complexity-theoretic properties

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.34

1 Introduction

Recently Bana and Comon in [3] – with others in followup papers [1, 4] – used a non-Tarskian semantics for first-order logic to formalize security properties of cryptographic primitives and of protocols, and to deduce the latter from the former with first-order deduction rules. By "non-Tarskian" we mean that disjunction, negation, and the existential quantifier were interpreted in a way different from usual, but nevertheless first-order deduction rules turned out to be sound for this interpretation. The non-Tarskian semantics emerged entirely naturally while they were trying to define what it means that a probabilistic polynomial-time attacker can compute a secret, what it means that it can either compute something (on certain random inputs) *or* something else (on other random inputs), and so on. Once they verified directly that despite the non-standard semantics first-order deduction rules hold, they realized that there might be some more general logical idea behind their result. And indeed, the result turned out (see [4]) to be a special case of Fitting's embedding of classical logic in S4 [7]. Fitting's embedding consists of applying $\Box\Diamond$ recursively on each sub-formula of a first-order formula. The resulting formula is deducible in first-order S4 if and only if the original is deducible in classical first-order logic. A non-Tarskian semantics to first-order

* Partially supported by ERC Consolidator Grant CIRCUS (683032).
† Partially supported by MEXT-Grant-in-Aid for Scientific Research on Innovative Areas (Grant number 23120002) and MEXT-JSPS Grant-in-Aid for Scientific Research (B) (Grant number 26284005). Also partially supported by the Next Generation Research Project Promotion Program of Keio University.

© Gergei Bana and Mitsuhiro Okada;
licensed under Creative Commons License CC-BY

logic is obtained by composing Fitting's embedding with a standard Kripke-semantics to first-order S4, and such is the one given by Bana and Comon. Fitting's work is in turn closely related to Cohen's forcing [5], as Fitting and Smullyan stated in [7, 15]. Forcing, and Fitting's technique allow to generate new models for first-order logic from given ones. The idea of generating new models is used in a special way in computer security: namely, generating attacker models for security protocols that are executable for probabilistic polynomial-time attackers. In this paper, we shall call a model that is generated by Fitting's embedding, a Fitting-twisted model.

The aim of this paper is to look (in a self-contained manner) at what aspects of the computability model considered in [3] are essential for the Fitting embedding to rise naturally, and explore a more general contexts in which the use of Fitting-twisted models may be the most suitable. One such aspect is the need for the following non-standard interpretation of disjunction and negation (and hence classical implication): Roughly speaking, imagine that there is a set W on which satisfaction $w \models \phi$ of first-order formulas is defined for each $w \in W$, when quantification runs over a single domain (variables in ϕ are interpreted over some domain $\mathcal{D}_w$ as usual in first-order logic, and we take $\mathcal{D}_w$ to be the same $\mathcal{D}$ for all $w \in W$)[1]. From this we want to define another satisfaction $W \models' \phi$ of the same first-order formulas. Instead of defining the satisfaction of a disjunction on W as usual so that either one of the disjuncts or the other is satisfied on W, it may be more natural (for example, as in case of the team semantics in [16]) to require that W can be split in two parts, on one of which one of the disjuncts is satisfied, and on the other, the other disjunct: $W \models' \phi_1 \vee \phi_2$ iff $W = W_1 \cup W_2$ and for both $i = 1, 2$, for all $w \in W_i$, $w \models \phi_i$. Negation on the other hand can mean that the negated predicate holds nowhere: $W \models' \neg\phi$ iff for all $w \in W$, $w \not\models \phi$. Then the first-order material implication $\phi \to \psi \equiv \neg\phi \vee \psi$ turns out to mean under $\models'$ that on the part where ϕ holds, ψ also must hold. This in itself would not require Fitting embedding, only an embedding of classical logic in S5 by applying $\Box$ recursively on each sub-formula (as touched upon in [7]) as $W \models' \phi$ would just mean $W \models' \phi$ if and only if $w \models \phi$ for all $w \in W$. The other aspect of computability in [3] that warrants the need for the Fitting-embedding to appear is the importance of random sets with "non-negligible" probabilities. Non-negligible sets can be looked at as generalized non-zero-measure sets. In computing, just as in measure theory, we are often not interested in what happens on negligible or measure-zero sets. So there is a notion of "significant" sets and "insignificant" sets, and we are only interested in what happens up to an error of insignificant sets. On insignificant sets, anything is allowed. When looking at implication, we do not want to mean that whenever the premise holds, the conclusion also holds, but we only want to mean that whenever the premise holds on a significant set, the conclusion also holds there with at most insignificant error. This makes it necessary to switch from W as the set of possible worlds to the significant sets of W as possible worlds, on which now we have a transitive accessibility relation: inclusion. This creates an S4 Kripke-model situation, leaving the S5 model.

In fact, in the case of computability, there is a further aspect that makes taking non-negligible sets to be the possible worlds necessary: There are important properties the satisfaction of which cannot be considered pointwise by $\models$ on W, but only on non-negligible sets. (This is similar to that in probability theory, conditionalization only makes sense on sets with non-zero measure. Or, in function analysis, differentiability only makes sense on open sets.) A single bit string is always computable from another bit string. However, when we talk about distributions of bit strings parametrized by some size, then we can

[1] This is sufficient for us, but Fitting's theorem works for varying domains as well.

ask if there exist algorithms within some complexity class that compute one parametrized distribution from another on a set that has non-negligible probability. The satisfaction of this property cannot be considered under $w \models$. This aspect reinforces the necessity to distinguish significant and insignificant sets.

Besides the applicability of the Fitting-embedding to the situation when we are interested in satisfaction of first-order properties on significant sets only, there may be another important aspect of this connection going in the other direction, from deduction system to semantics. This significant sets construction is also an extension of standard first-order Tarski semantics, just as the Fitting twist, and forcing. But it is less general, more concrete, and hence perhaps more intuitive. One wonders, can this significant sets construction be directly used for model construction in first-order logic to investigate consistency of formulas, other than for finding attacks to protocols, in situations more important for logic itself. We do not have such examples yet.

In Section 2 and 3 we motivate and build up the Fitting-twisted semantics. In this, we shall keep in mind the application to computer security, but we shall also consider other examples, conditional implication, and the ϵ-semantics of "typically" [11, 8]. In Section 4 we shall explain the connection with Fitting's embedding, and mention when and how first-order deduction is sound in the concrete examples of conditional implication and computer security.

2 Semantics for Enough-Certainty

Suppose that we have a set of possible worlds W. Suppose also that we are interested in first-order statements the terms of which are interpreted in a domain $\mathcal{D}$ and the satisfaction of its predicates are defined for all $w \in W$ with the single domain $\mathcal{D}$. Hence satisfaction of first-order formulas is defined for all $w \in W$ in the usual way. Now suppose also that we are interested in the satisfaction of the same formulas over subsets of W allowing some possible insignificant error. Think for example of measure theory: Measure zero sets are considered insignificant. Equality of functions over this measure space can be considered at each point, but it can also be considered over sets of the measure space, "almost everywhere", that is, except for a measure-zero set. We shall see more complex examples later. Motivated by this simple measure-theoretic example, we also allow sets of possible worlds for which we cannot tell if it is significant or insignificant.

Accordingly, consider the following definitions:

▶ **Definition 1.** Let W be a set, $\Sigma \subseteq 2^W$ be a σ-algebra, and let Σ_i be a non-trivial ideal of Σ, namely:

- if $S \in \Sigma_\mathrm{i}$, and $S' \in \Sigma$, and $S' \subseteq S$, then $S' \in \Sigma_\mathrm{i}$;
- if $S \in \Sigma_\mathrm{i}$, and $S' \in \Sigma_\mathrm{i}$, then then $S \cup S' \in \Sigma_\mathrm{i}$;
- $\Sigma_\mathrm{i} \neq \Sigma$.

We call Σ_i the set of insignificant sets and we call $\Sigma_\mathrm{s} := \Sigma \setminus \Sigma_\mathrm{i}$ the set of significant sets. We shall refer to elements of Σ as events. If the complement of S, $S^\perp \in \Sigma_\mathrm{i}$, then we call S a certain enough set. Note, the set of certain enough sets is the filter complementary to Σ_i.

We shall also use the term insignificant set implicitly meaning that it is an insignificant event. In other words, sub-events of insignificant events (sets) are insignificant, this is a rather obvious condition. It is also rather obvious that the full space W should be significant, that is, $\Sigma_\mathrm{i} \neq \Sigma$ should be required, otherwise we have nothing significant. We also assume that the union of two insignificant events are also insignificant; this is just an assumption that insignificant sets are indeed insignificant enough not to turn into significant ones so easily as taking finite unions.

To see more complex examples of insignificant sets than the measure zero sets, consider the following definitions.

Let $\mathbb{R}^+$ denote the non-negative real numbers. Let $\mathcal{C}$ be a convergence class in the following sense:

▶ **Definition 2** (Convergence Class)**.** We call $\mathcal{C}$, a set of $\mathbb{N} \to \mathbb{R}^+$ sequences a *convergence class*, if

- for each $s \in \mathcal{C}$, all subsequences of s are also in $\mathcal{C}$,
- for all $s \in \mathcal{C}$, $\lim_{i\to\infty} s_i = 0$;
- if $s \in \mathcal{C}$, and $s' : \mathbb{N} \to \mathbb{R}^+$, and for all $i \in \mathbb{N}$, $s'_i \leq s_i$, then $s' \in \mathcal{C}$;
- if $s, s' \in \mathcal{C}$, then $s + s' \in \mathcal{C}$.

Note that $\mathcal{C}$ is an ideal of sequences of the form $\mathbb{N} \to \mathbb{R}^+$ if the partial order $\leq$ on the set of such sequences is defined to hold if all elements of the sequence on the left are less than or equal to the corresponding elements of the sequence on the right. Note also, these conditions imply that if $s \in \mathcal{C}$, and $r \in \mathbb{R}^+$, then $r \cdot s \in \mathcal{C}$, because $r \leq n$ for some n natural, and then $r \cdot s \leq n \cdot s = s + \ldots + s$.

$\mathcal{C}$ can be for example the set of all positive sequences converging to 0, or the set of sequences super-polynomially converging to 0, and so on. In this latter case, the elements of $\mathcal{C}$ are called *negligible functions*. We shall define them later precisely.

▶ **Definition 3** (Parametric Probability Space)**.** Let Ω be a set of elementary events with $\Sigma \subseteq 2^\Omega$ a σ-algebra on Ω. For each $i \in \mathbb{N}$, let P_i be a (σ-additive) probability measure on Σ, and let $\mathcal{P}$ denote the sequence $(P_i)_{i\in\mathbb{N}}$. We call $(\Omega, \Sigma, \mathcal{P})$ a parametric probability space.

▶ **Definition 4** ($\mathcal{C}$-Asymptotically Possible/Impossible/Certain Sets)**.** Let $\mathcal{C}$ be a convergence class, and let $(\Omega, \Sigma, \mathcal{P})$ be a parametric probability space. We call a set $S \in \Sigma$ a $\mathcal{C}$-asymptotically possible set if $(P_i(S))_{i\in\mathbb{N}} \notin \mathcal{C}$ and we call it a $\mathcal{C}$-asymptotically impossible set if the sequence $(P_i(S))_{i\in\mathbb{N}} \in \mathcal{C}$. Let $\mathcal{S}_{\mathcal{C}\text{-p}}$ denote the set of $\mathcal{C}$-asymptotically possible elements of Σ, and let $\mathcal{S}_{\mathcal{C}\text{-i}}$ denote the set of $\mathcal{C}$-asymptotically impossible elements of Σ. We call a set S $\mathcal{C}$-asymptotically certain if its complement, $S^\perp$ is $\mathcal{C}$-asymptotically impossible, that is, if $(1 - P_i(S))_{i\in\mathbb{N}} \in \mathcal{C}$.

Clearly, a $\mathcal{C}$-asymptotically impossible set is a special insignificant set, a $\mathcal{C}$-asymptotically possible set is a significant set and a $\mathcal{C}$-asymptotically certain set is a certain enough set.

▶ **Example 5.** The simplest examples of parametric probability spaces are when all P_i coincide. In that case there is only one probability, and it can only converge to 0 if it is 0. Accordingly, in that special case, the asymptotically impossible sets are the measure zero sets, the asymptotically certain sets are the sets with measure 1. ◀

▶ **Example 6.** We can imagine the sequence of probability distributions $P_1, P_2, \ldots$ as change corresponding to time. For example, when one considers a Markov chain. In that case, a $\mathcal{C}$-asymptotically impossible set is one for which the probability goes to 0 in a way characterized by $\mathcal{C}$ as time progresses. ◀

▶ **Example 7.** The example which we elaborate mostly in this paper is when the sequence $P_1, P_2, \ldots$ corresponds to some size parameter. This is the case for the mathematical formalization of hardness assumptions such as the discrete logarithm problem, Diffie-Hellman problem etc. In this case, we consider $\{0,1\}^*$ with Σ_0 the σ-algebra generated by those sets for which the first n bits are fixed. For example, elements of the form 01001ω where

$\omega \in \{0,1\}^*$ make up such a set, with $n = 5$. Let P^c be the probability distribution that gives $1/2^n$ on such sets: fair coin tosses. Let

$$\Omega^c := \mathbb{N} \times \{0,1\}^*$$

with the product σ-algebra Σ^c (where $2^{\mathbb{N}}$ is taken on $\mathbb{N}$ and Σ_0 on $\{0,1\}^*$). That is, Σ^c is generated by sets of the form $S_1 \times S_2$ where $S_1 \in 2^{\mathbb{N}}$ and $S_2 \in \Sigma_0$. Clearly, each $S \in \Sigma^c$, can be written as

$$S = \bigcup_{i=0}^{\infty} \{i\} \times S_i$$

for some $S_i \in \Sigma_0$. We define $\mathcal{P}^c = (P_i^c)_{i \in \mathbb{N}}$ such that for all $S = \bigcup_{i=0}^{\infty} \{i\} \times S_i$,

$$P_i^c(S) := P^c(S_i).$$

That is, $P_i^c(S)$ gives the probability of the i'th section of S according to P^c. Let $\mathcal{C}^c$ denote the set of those sequences that go to 0 faster than the inverse of any polynomial: for each $s \in \mathcal{C}^c$ and $n \in \mathbb{N}$, there is an $i \in \mathbb{N}$ such that for all $j \in \mathbb{N}$ with $j \geq i$, we have $s_j \leq 1/j^n$. Then the $\mathcal{C}^c$-asymptotically impossible elements of Σ_{i}^c are called *negligible events* (or sets with negligible probability) in the literature and $\mathcal{C}^c$-asymptotically possible sets in Σ_{s}^c are called *non-negligible*.

▶ **Example 8.** Another example comes from natural language, to interpret the meaning of "typically". For example, "Typically birds can fly". Sequences of probability distributions were used for example in [11, 8] for giving semantics to such statements (ϵ-semantics). In this case, $\mathcal{C}$ contains all sequences converging to 0. Something typically holds, if it holds outside an asymptotically impossible set.

Consider now the first-order signature $(\mathfrak{f}, \mathfrak{p})$ with $\mathfrak{f}$ a set of function symbols and $\mathfrak{p}$ a set of predicates. Let $A_{(\mathfrak{f},\mathfrak{p})}$ denote the atomic formulas built on $(\mathfrak{f}, \mathfrak{p})$ with some countable set of variables. Let $\Phi_{(\mathfrak{f},\mathfrak{p})}$ denote the first order formulas built on $(\mathfrak{f}, \mathfrak{p})$ with the same countable set of variables. Let $\mathcal{D}$ be a domain of interpretation, that is, terms are interpreted as elements of $\mathcal{D}$, and predicates are interpreted as relations on $\mathcal{D}$. Let W be a set of possible worlds with the fixed domain $\mathcal{D}$; each $w \in W$ defines an interpretation of the function symbols and the predicates on $\mathcal{D}$. Let $\mathcal{V}$ denote the set of valuations of variables in $\mathcal{D}$. Then a relation $\models$ is defined on $\mathcal{V} \times W \times \Phi$ the standard first-order way; for all $V \in \mathcal{V}$, $w \in W$, and $\phi \in \Phi_{(\mathfrak{f},\mathfrak{p})}$, either $V, w \models \phi$ or $V, w \not\models \phi$.

In the following, we shall also assume that a σ-algebra Σ and a set of insignificant events Σ_{i} and hence significant events Σ_{s} are also defined on W. Let us call such a $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ a *possible world model with significant events*.

Coming back to our question at the beginning of this section, using the possible-world-wise satisfaction $w \models \phi$, how should we define $W \models' \phi$? Or, more generally, how should we define $S \models' \phi$ for a set $S \in \Sigma$ and $\phi \in \Phi_{(\mathfrak{f},\mathfrak{p})}$ using the pointwise satisfaction $w \models \phi$? One obvious answer is the following.

▶ **Definition 9** (Absolute Certainty Semantics)**.** Let $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ be a possible world model with significant events. Let $\mathcal{V}$ denote the set of valuations of variables in $\mathcal{D}$. We can define the relation $\models_{\mathrm{AC}}$ on $\mathcal{V} \times \Sigma \times A_{(\mathfrak{f},\mathfrak{p})}$:

$$V, S \models_{\mathrm{AC}} \phi \quad \Leftrightarrow \quad \forall w \in S.\ (V, w \models \phi)\,.$$

Then $\models_{\text{AC}}$ can be extended to $\mathcal{V} \times \Sigma \times \Phi_{(\mathfrak{f},\mathfrak{p})}$ as usual for first-order semantics. We call this *absolute certainty semantics* of the first-order formulas as satisfaction is required everywhere on S.

Here, and in what follows we use bold face $\forall$, $\exists$, $\wedge$, $\vee$, $\neg$ for meta expressions, while we use the standard ones for our first order formulas.

But in this work, we shall not care what happens on insignificant sets. Just as in measure theory, one does not care what happens on 0-measure sets, and in complexity theory we do not care what happens on negligible sets. We only care about properties satisfied up to an error of insignificant set. Then it makes sense to define the following satisfaction for any $S \in \Sigma_{\text{s}}$.

▶ **Definition 10** (Enough-Certainty Semantics)**.** Let $(\mathcal{D}, W, \Sigma, \Sigma_{\text{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ be a possible world model with significant events. Let $\mathcal{V}$ denote the set of valuations of variables in $\mathcal{D}$. We can define the relation $\models_{\text{EC}}$ on $\mathcal{V} \times \Sigma_{\text{s}} \times A_{(\mathfrak{f},\mathfrak{p})}$:

$$V, S \models_{\text{EC}} \phi \quad \Leftrightarrow \quad \exists S' \in \Sigma_{\text{s}}.\ (S' \subseteq S \ \ \wedge \ \ S \setminus S' \in \Sigma_{\text{i}} \ \ \wedge \ \ V, S' \models_{\text{AC}} \phi)\,.$$

Again, $\models_{\text{EC}}$ can be extended to $\mathcal{V} \times \Sigma_{\text{s}} \times \Phi_{(\mathfrak{f},\mathfrak{p})}$ as usual for first-order semantics. We call this *enough-certainty semantics* of the first-order formulas as satisfaction is required up to an error of an insignificant subset $S \setminus S'$ of S.

▶ **Example 11.** Given a parametrized probability space $(\Omega, \Sigma, \mathcal{P})$, let the $\mathcal{C}$-asymptotically impossible sets be the insignificant sets. For two random variables $X, Y : \Omega \to \mathbb{R}$, for each ω, we can define $\omega \models X = Y$ iff $X(\omega) = Y(\omega)$. Similarly, we can define $\omega \models X \leq Y$ iff $X(\omega) \leq Y(\omega)$. According to the above definition, $\Omega \models_{\text{AC}} X = Y$ iff $X(\omega) = Y(\omega)$ for all ω. Similarly for $\leq$. Also, $\Omega \models_{\text{EC}} X = Y$ if $X(\omega) = Y(\omega)$ for all ω except possibly on a $\mathcal{C}$-asymptotically impossible set. Moreover, for other two random variables $U, W : \Omega \to \mathbb{R}$, we have that $\Omega \models_{\text{AC}} X = Y \vee U = W$ if either $X = Y$ on all of Ω, or $U = W$ on all of Ω.

Now this may not be the best definition on a probability field. When we say that either $X = Y$ or $U = W$, we might want to mean that on a part of Ω (that is, for certain randomness) $X = Y$ holds, and on its complement $U = W$ holds, not necessarily one of them everywhere. The same can be said about $\models_{\text{EC}}$ up to a $\mathcal{C}$-asymptotically impossible sets.

▶ **Example 12.** Coming back to Example 8, the semantics of "typically, birds can fly or birds are sick" should not be the same as "typically birds can fly or typically birds are sick". Similarly, "typically monkeys do not fly" is not the same as "it is not true that typically monkeys fly". The latter would allow monkeys to fly often enough, maybe in 1/3 of the cases, but not typically. The former only allows monkeys to fly very rarely. On the other hand, "typically, birds can fly or birds are sick" can be the same as "typically, typically birds can fly or typically birds are sick" if we assume (and we do) that the union of two atypical sets is also atypical.

The the above examples show that for a $(\mathcal{D}, W, \Sigma, \Sigma_{\text{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ possible world model, although $\models_{\text{AC}}$ and $\models_{\text{EC}}$ interpret connectives and quantifiers as usual in first-order logic, they are not necessarily what we expect from a good semantics. It is intuitive to define another semantics as well. In this new semantics, for an $S \in \Sigma_{\text{s}}$, $S \models \phi_1 \vee \phi_2$, that is, on S, either ϕ_1 or ϕ_2 is satisfied, could mean that S can be divided into (or covered by) two parts, one on which ϕ_1 holds, one on which ϕ_2 holds. As the existential quantifier is a generalized disjunction, the analogous definition for the semantics of $\exists$ would be that S can be covered by elements in Σ_{s} on each of which there is a witness. Coming back to our Example 11,

let us consider $X = Y \vee X \neq Y$. Having accepted that $\vee$ is interpreted as a division of S into two parts, one on which $X = Y$ holds, another on which $X \neq Y$ holds, clearly, $X \neq Y$ should be interpreted as $X = Y$ holds nowhere on the part where $X \neq Y$ holds.

One can also think of coin tosses: The statement that the result of a coin toss is either heads or tails means that on certain part of the underlying probability field, the coin comes down heads, on the other part it comes down tails. There is no other option. When we say that the result of throwing the dice is either 1 or not 1, we mean that on a part of the underlying probability field, it is 1, on the other part it is nowhere 1.

Accordingly, adding that we do not care what happens on insignificant sets, we need the following:

▶ **Definition 13** (Covering Enough-Certainty Semantics)**.** Let $(\mathcal{D}, W, \Sigma, \Sigma_s, \mathfrak{f}, \mathfrak{p}, \models)$ be a possible world model with significant events. Let $\mathcal{V}$ denote the set of valuations of variables ranging over $\mathcal{D}$. We define the relation $\models_{\text{CEC}}$ on $\mathcal{V} \times \Sigma_s \times \Phi_{(\mathfrak{f},\mathfrak{p})}$ in the following way:

- If $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$, then $V, S \models_{\text{CEC}} \phi$ iff $V, S \models_{\text{EC}} \phi$.
- The semantics of $\wedge$ and $\forall$ are the usual ones.
- $V, S \models_{\text{CEC}} \phi_1 \vee \phi_2 \Leftrightarrow \exists S_1, S_2 \in \Sigma_s.\big(S \setminus (S_1 \cup S_2) \in \Sigma_i \ \wedge\ V, S_1 \models_{\text{CEC}} \phi_1 \ \wedge\ V, S_2 \models_{\text{CEC}} \phi_2\big)$
- $V, S \models_{\text{CEC}} \neg\phi \Leftrightarrow \forall S' \in \Sigma_s.\big(S' \subseteq S \Rightarrow V, S' \not\models_{\text{CEC}} \phi\big)$
- $V, S \models_{\text{CEC}} \phi_1 \to \phi_2 \Leftrightarrow \forall S' \in \Sigma_s.\big(S' \subseteq S \ \wedge\ V, S' \models_{\text{CEC}} \phi_1 \Rightarrow V, S' \models_{\text{CEC}} \phi_2\big)$
- $V, S \models_{\text{CEC}} \exists x\phi[x] \Leftrightarrow \forall S' \in \Sigma_s.\Big(S' \subseteq S \Rightarrow$
  $\exists \mathcal{S} \subseteq \Sigma_s.\Big(S' \setminus \bigcup_{S'' \in \mathcal{S}} S'' \in \Sigma_i$
  $\wedge\ \forall S'' \in \mathcal{S}.\big(\exists V' \in \mathcal{V} \text{ differing from } V \text{ only on } x.\ (V', S'' \models_{\text{CEC}} \phi[x])\big)\Big)\Big)$

We call this *covering enough-certainty semantics* of the first-order formulas.

▶ Remark. Note, the first thought would be to define satisfaction of the existential quantifier as

$$V, S \models_{\text{CEC}} \exists x\phi[x] \Leftrightarrow \exists \mathcal{S} \subseteq \Sigma_s.\Big(S \setminus \bigcup_{S'' \in \mathcal{S}} S'' \in \Sigma_i \wedge \forall S'' \in \mathcal{S}.\big(\exists V' \in \mathcal{V} \text{ differing from } V \text{ only on } x.\ (V', S'' \models_{\text{CEC}} \phi[x])\big)\Big).$$

However, this is not good, because the insignificant parts in the S''s where $\phi[x]$ is not satisfied, may add up to a significant subset of S. Hence the above definition.

If $\mathcal{D}$ only has countably many elements, and if $\{w \in W : w \models \phi\}$ is measurable for all $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$, then although the semantics of compound formulas with respect to $\models_{\text{CEC}}$ is not defined the usual Tarski way, first order logical deduction rules are valid with respect to $\models_{\text{CEC}}$. We shall see the proof of this in Section 4. In particular, when these conditions are satisfied, the usual distributivity holds for disjunction and conjunction, $\neg\exists$ is the same as $\forall\neg$ and so on. We shall also see in Section 4 that it is possible to give a better general semantics, namely Definition 21, for which first order deduction rules work even when $\mathcal{D}$ is not countable or when $\{w \in W : w \models \phi\}$ is not measurable.

▶ Remark. Note that for ϕ without quantifiers, we have

$$V, S \models_{\text{CEC}} \phi \Leftrightarrow \exists S' \in \Sigma_s.\big(S \setminus S' \in \Sigma_i \ \wedge\ V, S' \models_{\text{AC}} \phi\big) \tag{1}$$

For any ϕ without quantifiers, let

$$[\phi]_V := \big\{w \big| w \in W \ \wedge\ V, w \models \phi\big\}$$

When ϕ and ψ have no quantifiers, and $[\psi]_V$ (and hence $[\neg\psi]_V$) and $[\phi]_V$ are measurable, and in that case,

$$V, S \models_{\mathrm{CEC}} \phi \to \psi \quad \Leftrightarrow \quad [\neg\psi]_V \wedge [\phi]_V \in \Sigma_{\mathrm{i}} \tag{2}$$

also holds. In other words, on $[\phi]_V$, ψ may fail only on an insignificant set.

▶ **Example 14.** When insignificant sets are the $\mathcal{C}$-asymptotically impossible sets of a parametric probability field $(\Omega, \Sigma, \mathcal{P})$, then for ϕ and ψ without quantifiers, if $[\phi]_V$ and $[\psi]_V$ are measurable, then we have

$$V, S \models_{\mathrm{CEC}} \phi \to \psi \quad \Leftrightarrow \quad \Big(1 - P_i\big(\ [\psi]_V \ \big|\ [\phi]_V\ \big)\Big)_{i\in\mathbb{N}} \in \mathcal{C} \tag{3}$$

where $P_i(\ [\psi]_V \mid [\phi]_V\ )$ is the probability of $[\psi]_V$ conditioned on $[\phi]_V$. This is again because $V, S \models_{\mathrm{CEC}} \neg\phi\vee\psi$ means exactly that on the part of S where ϕ holds, ψ also has to hold except for an $\mathcal{C}$-asymptotically impossible set, which exactly means that $(1-P_i(\ [\psi]_V \ \big|\ [\phi]_V\ ))_{i\in\mathbb{N}} \in \mathcal{C}$. In other words, $[\psi]_V$ is $\mathcal{C}$-asymptotically certain conditioned on $[\phi]_V$.

For this conditional implication however, another connective $\to'$ is introduced in [11, 8] with the above semantics in (3). That is, a conditional implication that holds "typically". One of the conclusions of this paper in Example 26 will be that when the sets of the form $[\phi]_V$ are measurable for atomic formulas, and we are only interested in what is satisfied on $\mathcal{C}$-asymptotically possible sets, then we do not need to introduce a new implication for this, the usual one together with the Fitting twist provides this semantics for $\phi \to \psi$ where ϕ and ψ are formulas without quantifiers, and classical first-order deduction rules are sound.

▶ **Example 15.** Consider $[0, 1] \subset \mathbb{R}$ with Lebesgue measurability and the uniform distribution. For the random variable X that is 0 before $1/2$ and 1 from $1/2$, according to the above definition, $X = 0 \vee X = 1$ is satisfied. Which is intuitive. Let Y be the random variable for which $Y(\omega) = n$ for $1/(n + 1) < \omega \leq 1/n$. Then according to $\models_{\mathrm{CEC}}$, it is satisfied that "There is a constant random variable Z such that $Z = Y$". That is because Y is locally constant, we can cover the space with measurable sets on each of which Y is constant. If we make equality a congruence, then this constant predicate cannot distinguish between globally constant and locally constant notions.

The same statement "There is a constant random variable Z such that $Z = W$" fails to hold for $W(\omega) = \omega$. This shows that it is easy to come up with formulas for which it is not good to define satisfaction on S by pointwise satisfaction for quantifiers: For this W, obviously, all ω satisfies that there is a constant c such that $W(\omega) = c$. So had we defined $\models_{\mathrm{CEC}}$ such that (1) hold for ϕ with existential quantifier as well, S would satisfy for any random variable that there is a constant equaling the random variable, which we do not want.

▶ **Example 16.** In case of continuous distributions, if we are only interested in satisfaction of properties up to an error of zero probability – which is the standard in measure theory and probability theory – then it is useless to require existence with satisfaction at each point when the points have zero measure. A formula of the sort $\exists f$. (f satisfies some property except on a set with zero probability) cannot be made sense pointwise on each ω for continuous distributions. Only on sets with non-zero probability can we have statements that hold up to zero-probability.

▶ **Example 17.** Considering "typically", similarly to up to zero probability, we might need more than a single point to evaluate statements. Such are those that have the form "typically

exists ... that typically ... ". For example, "Typically there are chain shops that typically sell pens." meaning that typically in every region there is a chain shop that typically (in that region) sells pens. It is possible that there are some atypical regions where there are no chain shops selling pens. It is also possible that a chain shop that typically sells pens in a region does not do so in another region. In that other region another chain shop sells it. It is clear that this existence cannot be defined pointwise.

In [11], the authors define a conditional entailment $\phi \mid\!\sim \psi$ for atomic formulas exactly as $(1 - P_i(\ [\psi]_V \mid [\phi]_V\ ))_{i\in\mathbb{N}} \in \mathcal{C}$, which is the same what we have in Equation (2). As the authors note, $\phi \mid\!\sim \psi$ does not imply $\phi \wedge \phi' \mid\!\sim \psi$ in general. For example, "typically birds fly" does not mean that "typically penguin birds fly". That is because penguins are atypical. However, if we are only interested in statements on typical sets, then, as we shall see, first order deduction rules can be made to work even when the domain is not countable. In that case, $\phi \wedge \phi' \mid\!\sim \psi$ is implied because $\phi \wedge \phi'$ is atypical, that is, insignificant.

▶ Remark. Note that when W has only one element (and hence the probability distributions all agree), $\models_{\mathrm{AC}}$, $\models_{\mathrm{EC}}$, $\models_{\mathrm{CEC}}$ all coincide with $\models$. In other words, $\models_{\mathrm{AC}}$, $\models_{\mathrm{EC}}$, $\models_{\mathrm{CEC}}$ can be considered extensions of the Tarski-semantics for first-order logic.

The three semantics of the previous section are defined using a basic satisfaction notion $\models$ on possible worlds $w \in W$. Sometimes we need something more general, when the satisfaction of some predicate does not come from pointwise satisfaction.

▶ **Definition 18** (Significant Sets Semantics)**.** Let $(W, \Sigma, \Sigma_{\mathrm{s}})$ such that W is a set, Σ is a σ-algebra on W, and Σ_{s} is a subset of significant sets in σ. Let $(\mathfrak{f}, \mathfrak{p})$ be a first-order language. Let $\mathcal{D}$ be a domain for interpretation. Let $\mathcal{V}$ denote the set of valuations of first-order variables with codomain $\mathcal{D}$. Suppose for each Σ_{s}, the predicates are interpreted on $\mathcal{D}$, and hence a relation $\models_{\mathrm{SiS}}$ on $\mathcal{V} \times \Sigma_{\mathrm{s}} \times A_{(\mathfrak{f},\mathfrak{p})}$ is given, which we call a *significant sets semantics*. $\models_{\mathrm{SiS}}$ can be extended to $\mathcal{V} \times \Sigma_{\mathrm{s}} \times \Phi_{(\mathfrak{f},\mathfrak{p})}$ as usual for first-order semantics.

Clearly, $\models_{\mathrm{AC}}$ and $\models_{\mathrm{EC}}$ are special cases of $\models_{\mathrm{SiS}}$ for $\Phi_{(\mathfrak{f},\mathfrak{p})}$.

3 Kripke Semantics and Fitting-Twisted Semantics for Enough-Certainty

In this section, we exploit the fact that there is a transitive reachability relation on Σ_{s} such that S' is reachable from S if and only if $S' \subseteq S$. Let $\Phi^K_{(\mathfrak{f},\mathfrak{p})}$ denote the set of modal formulas with the addition of $\Box$ and $\Diamond$ operators to $\Phi_{(\mathfrak{f},\mathfrak{p})}$.

▶ **Definition 19** (Kripke Absolute Certainty, Kripke Enough Certainty, Kripke Significant Sets Semantics)**.** With $\subseteq$ as a reachability (also called accessibility) relation, all of $\models_{\mathrm{AC}}$, $\models_{\mathrm{EC}}$, and $\models_{\mathrm{SiS}}$ can be extended to $\models_{\mathrm{KAC}}$, $\models_{\mathrm{KEC}}$, and $\models_{\mathrm{KSiS}}$ respectively on $\mathcal{V} \times \Sigma_{\mathrm{s}} \times \Phi^K_{(\mathfrak{f},\mathfrak{p})}$ interpreting $\Box$ and $\Diamond$ as usual for first-order Kripke semantics. This makes three different S4 semantics because the set-inclusion is reflexive and transitive.

$\models_{\mathrm{KAC}}$, $\models_{\mathrm{KEC}}$, are special instances of $\models_{\mathrm{KSiS}}$. We shall use these Kripke semantics later to obtain the Fitting-twisted first-order semantics.

▶ **Example 20.** Now we show an example in which the interpretation of a predicate cannot be defined point-wise on each $w \in W$. We also use this example to motivate our targeted semantics. Bana and Comon in [3] presented a technique for the verification of trace properties of cryptographic protocols. They used a predicate $t_1, \ldots, t_n \rhd t$ with the intuitive meaning that an attacker can compute t from $t_1, \ldots, t_n$ with a probabilistic polynomial-time

algorithm, while $t_1, \ldots, t_n \not\rhd t$ (that is, $\neg(t_1, \ldots, t_n \rhd t)$) means intuitively that there is no such algorithm. When t is a nonce N,[2] and $t_1, \ldots, t_n$ represent the messages sent by the honest agents, $t_1, \ldots, t_n \not\rhd N$ means that N remains secret: there is no probabilistic polynomial-time algorithm that computes it from the messages of the honest agents. But when they wanted to make this intuition formal, it turned out to be a difficult task. They tried several possibilities first, both for the interpretation of $t_1, \ldots, t_n \rhd t$ and various non-Tarskian interpretations of connectives, which all failed before arriving to the final definitions. This process was never actually published, but here we do that to illustrate how the non-Tarskian semantics naturally emerges in security. In fact the one presented in [3] still worked only for a fragment of the formulas, the final correct definitions only appeared in eprint [2] and in subsequent works such as [4].

Take $(\Omega^c, \Sigma^c, \Sigma^c_{\mathrm{s}}, \mathcal{P}^c)$ as in Example 7. We take the domain $\mathcal{D}^c$ to be probabilistic polynomial-time (PPT) algorithms such that for each $a \in \mathcal{D}^c$ the input is of the form $(1^\eta, \omega)$, where $\eta \in \mathbb{N}$, $\omega \in \{0,1\}^*$, and 1^η means a bit string with η many 1's. That is, the inputs of a are in Ω^c and the first component of elements of Ω^c are fed to a in the form of a string of 1's. The requirement that a is PPT means that a has to stop in polynomial number of steps $p(\eta)$ as a function of η. This means that a can only use an initial section of ω, its second input, namely at most $p(\eta)$ number of random bits.

The interpretations of terms $t_1, \ldots, t_n, t$ take values in this $\mathcal{D}^c$. For a $V \in \mathcal{V}$ valuation of variables in $\mathcal{D}$, let $[\![t_1]\!]_V, \ldots, [\![t_n]\!]_V, [\![t]\!]_V$ denote the interpretations in $\mathcal{D}$. How should the semantics of $t_1, \ldots, t_n \rhd t$ be defined to correspond to the intuition that a PPT attacker can compute t from $t_1, \ldots, t_n$? That is, taking an $S \in \Sigma^c_{\mathrm{s}}$, what should be the definition of $V, S \models^c t_1, \ldots, t_n \rhd t$? The first thought would be to say that $V, S \models^c t_1, \ldots, t_n \rhd t$ if and only if there is a PPT algorithm $\mathcal{A}$ that computes $[\![t]\!]_V$ from $[\![t_1]\!]_V, \ldots, [\![t_n]\!]_V$ except on a negligible set. That is, $\mathcal{A}([\![t_1]\!]_V(1^\eta, \omega), \ldots, [\![t_n]\!]_V(1^\eta, \omega), \omega) = [\![t]\!]_V(1^\eta, \omega)$ for all (η, ω), except on a negligible set of Ω^c (making sure that the bits in ω used by $\mathcal{A}$ in its last argument are disjoint from those used by $[\![t_1]\!]_V, \ldots, [\![t_n]\!]_V, [\![t]\!]_V$). This however does not work. The reason is that if there are two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, and $\mathcal{A}_1$ computes the RHS of $\rhd$ from the LHS of $\rhd$ for parts of the (η, ω) and $\mathcal{A}_2$ computes the RHS from the LHS for the rest of the (η, ω) pairs except maybe on a negligible set, that is a win for the attacker as well, although there may not be a single algorithm for the whole Ω^c. Furthermore, when in computer security we prove that $t_1, \ldots, t_n \not\rhd t$, we want the meaning to be such that there is no $\mathcal{A}$ PPT algorithm that computes the RHS from the LHS on a non-negligible set.

So let us fix the meaning of $V, S \models^c t_1, \ldots, t_n \not\rhd t$ first. We shall say that $V, S \models^c t_1, \ldots, t_n \not\rhd t$ if and only if for all $\mathcal{A}$ PPT, and all $S' \in \Sigma^c_{\mathrm{s}}$ with $S' \subseteq S$, there is an $(\eta, \omega) \in S'$ such that $\mathcal{A}([\![t_1]\!]_V(1^\eta, \omega), \ldots, [\![t_n]\!]_V(1^\eta, \omega), \omega) \neq [\![t]\!]_V(1^\eta, \omega)$. Note that in fact the inequality must be true for all elements in S' except for some with negligible probability, otherwise there would be a non-negligible subset of S' where the equality would be satisfied, that would contradict this definition. Hence, $V, S \models^c t_1, \ldots, t_n \not\rhd t$ if and only if for all $\mathcal{A}$ PPT, the set

$$\{(\eta, \omega) \in S : \mathcal{A}([\![t_1]\!]_V(1^\eta, \omega), \ldots, [\![t_n]\!]_V(1^\eta, \omega), \omega) = [\![t]\!]_V(1^\eta, \omega)\}$$

is negligible.

So then if we want the negation to be standard, $V, S \models^c t_1, \ldots, t_n \rhd t$ should be defined such that there is a PPT algorithm $\mathcal{A}$ that can compute the RHS from the LHS on some S' non-negligible subset of S. But the problem is that this definition has bad properties! For

[2] A nonce in security is a freshly, randomly generated bit string that can only be guessed with negligible probability

example $V, S \models^c t_1, \ldots, t_n \rhd t \wedge t_1, \ldots, t_n \rhd t'$ would not imply that $V, S \models^c t_1, \ldots, t_n \rhd (t, t')$, because the non-negligible set on which t can be computed may be disjoint from the non-negligible set on which t' can be computed, and then there is no non-negligible set on which both t and t' can be computed from the LHS of $\rhd$. Hence it looks like we either have to give up the standard Tarski definition of the semantics of negation, or we have to deal with bad properties. Giving up Tarski definition of the semantics of negation, likely means giving up the use of first-order deduction. But as we shall see this is not the case! We can give up Tarskian semantics and still keep first-order deduction.

So let us keep the definition of $V, S \models^c t_1, \ldots, t_n \not\rhd t$, give up Tarski's semantics for negation, and continue thinking what $V, S \models^c t_1, \ldots, t_n \rhd t$ should be. The next idea is that we require the RHS to be computable from the LHS except maybe for negligible subset of S not by one algorithm, but many. That is, maybe we should define $V, S \models^c t_1, \ldots, t_n \rhd t$ so that S can be covered by non-negligible sets $S_i \in \Sigma^c_{\mathrm{s}}$ such that $S \setminus \bigcup_i S_i \in \Sigma^c_{\mathrm{i}}$, and for each S_i there is an algorithm $\mathcal{A}_i$ that computes the RHS from the LHS. This however is still problematic, $V, S \models^c t_1, \ldots, t_n \rhd t \wedge t_1, \ldots, t_n \rhd t'$ still would not imply $V, S \models^c t_1, \ldots, t_n \rhd (t, t')$. That is because although the sets on which both t and t' can be computed, that is, the intersections of the covering for t and the covering for t' still covers S except maybe for negligible probability, but those intersections may all be negligible!

So let us try to make the definition of $V, S \models^c t_1, \ldots, t_n \rhd t$ stronger than just a covering. Namely, for each covering of S, there is a refinement covering such that on this new covering there is an algorithm for each covering set that computes the RHS from the LHS on that set. This is equivalent with the following: $V, S \models^c t_1, \ldots, t_n \rhd t$ if and only if for all $S' \in \Sigma^c_{\mathrm{s}}$, $S' \subseteq S$, there is an $S'' \in \Sigma^c_{\mathrm{s}}$, $S'' \subseteq S'$, and an algorithm $\mathcal{A}$ such that for all $(\eta, \omega) \in S''$, $\mathcal{A}(\llbracket t_1 \rrbracket_V(1^\eta, \omega), \ldots, \llbracket t_n \rrbracket_V(1^\eta, \omega), \omega) = \llbracket t \rrbracket_V(1^\eta, \omega)$ (where $\mathcal{A}$ uses fresh part of ω). It is easy to check that $V, S \models^c t_1, \ldots, t_n \rhd t \wedge t_1, \ldots, t_n \rhd t'$ does imply $V, S \models^c t_1, \ldots, t_n \rhd (t, t')$, because for all non-negligible $S' \subseteq S$, there is a non-negligible $S'' \subseteq S'$ on which t can be computed, and S'' has a non-negligible subset S''' on which t' can be computed. Since on S''' both t and t' can be computed, S' has such a subset S''', and $V, S \models^c t_1, \ldots, t_n \rhd (t, t')$ is satisfied.

In the previous paragraph, we implicitly assumed that for conjunction, $V, S \models^c t_1, \ldots, t_n \rhd t \wedge t'_1, \ldots, t'_n \rhd t'$ is defined as usual. Let us keep this definition.

Now how should we define $V, S \models^c t_1, \ldots, t_n \rhd t \vee t'_1, \ldots, t'_n \rhd t'$? The first idea would of course be that S can be covered by two sets S_1 and S_2 such that $V, S_1 \models^c t_1, \ldots, t_n \rhd t$ and $V, S_2 \models^c t'_1, \ldots, t'_n \rhd t'$. But there is a better one. We have already defined negation and conjunction, so let us try that $V, S \models^c t_1, \ldots, t_n \rhd t \vee t'_1, \ldots, t'_n \rhd t'$ if and only if $V, S \models^c \neg(\neg t_1, \ldots, t_n \rhd t \wedge \neg t'_1, \ldots, t'_n \rhd t')$. By the definition of the interpretation of negation and of conjunction is easy to check that this turns out to be the following: $V, S \models^c t_1, \ldots, t_n \rhd t \vee t'_1, \ldots, t'_n \rhd t'$ if and only for all non-negligible set $S' \subseteq S$, there is a non-negligible $S'' \subseteq S'$ such that either $V, S'' \models^c t_1, \ldots, t_n \rhd t$ or $V, S'' \models^c t'_1, \ldots, t'_n \rhd t'$. This definition is not so far from the above splitting into S_1 and S_2, except that the unions of the sets on which one disjunct is satisfied and the unions of those on which the other may not be in Σ.

With a bit of similar considerations, defining the interpretation of $\forall$ as usual, and then the interpretation of $\exists$ through $\forall$ and negation, we arrive at a similar definition. These considerations motivate our next definition.

Motivated by the above example, we introduce the following notion, named after Melvin Fitting (it will be clear in Section 4.1 why):

▶ **Definition 21** (Fitting Twisted Enough-Certainty Semantics). Let $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \models_{\mathrm{SiS}})$ be a significant sets semantics for $\Phi_{(\mathfrak{f},\mathfrak{p})}$. Let $\mathcal{V}$ denote the set of valuations of first-order variables

with codomain $\mathcal{D}$. We can define the relation $\models_{\mathrm{FEC}}$ on $\mathcal{V} \times \Sigma_{\mathrm{s}} \times \Phi_{(\mathfrak{f},\mathfrak{p})}$ in the following way:

- $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$, then

$$V, S \models_{\mathrm{FEC}} \phi \;\Leftrightarrow \forall S' \in \Sigma_{\mathrm{s}}.\, \Big(S' \subseteq S \Rightarrow \exists S'' \in \Sigma_{\mathrm{s}}.\, (S'' \subseteq S' \wedge V, S'' \models_{\mathrm{SiS}} \phi)\Big)$$

- The semantics of $\wedge$ and $\forall$ are the usual ones.
- $V, S \models_{\mathrm{FEC}} \neg\phi \;\Leftrightarrow\; \forall S' \in \Sigma_{\mathrm{s}}.\, \Big(S' \subseteq S \Rightarrow V, S' \not\models_{\mathrm{FEC}} \phi\Big)$
- $V, S \models_{\mathrm{FEC}} \phi_1 \vee \phi_2$
  $\Leftrightarrow\; \forall S' \in \Sigma_{\mathrm{s}}.\, \Big(S' \subseteq S \;\Rightarrow\; \exists S'' \in \Sigma_{\mathrm{s}}.\, \big(S'' \subseteq S' \;\wedge\; (V, S'' \models_{\mathrm{FEC}} \phi_1 \;\vee\; V, S'' \models_{\mathrm{FEC}} \phi_2)\big)\Big)$
- $V, S \models_{\mathrm{FEC}} \exists x \phi[x]$
  $\Leftrightarrow\; \forall S' \in \Sigma_{\mathrm{s}}.\, \Big(S' \subseteq S$
  $\Rightarrow\; \exists S'' \in \Sigma_{\mathrm{s}}.\, \big(S'' \subseteq S' \wedge (\exists V' \in \mathcal{V}$ differing from V only on $x.\, (V', S'' \models_{\mathrm{FEC}} \phi[x]))\big)\Big)$

We call this *Fitting twisted enough-certainty semantics* of the first-order formulas.

In the next section we shall see that the deducibility/provability of classical first-order logic is sound with respect to this semantics.

▶ Remark. Let us introduce the abbreviation

$$\phi \to \psi \equiv \neg\phi \vee \psi$$

then it is easy to check that

- $V, S \models_{\mathrm{FEC}} \phi \to \psi \;\Leftrightarrow\; \forall S' \in \Sigma_{\mathrm{s}}.\, \Big(S' \subseteq S \wedge V, S' \models_{\mathrm{FEC}} \phi \Rightarrow V, S' \models_{\mathrm{FEC}} \psi\Big)$

In other words, $\phi \to \psi$ means that if ϕ is satisfied on a significant subset of S, then ψ also has to by satisfied there.

▶ **Proposition 22.** *Let $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \models_{SiS})$ and $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \models'_{SiS})$ be two significant sets semantics such that for each $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$ and $S \in \Sigma_{\mathrm{s}}$, $S \models_{SiS} \phi$ if and only if there is an $S' \in \Sigma_{\mathrm{s}}$ with $(S \setminus S') \cup (S' \setminus S) \in \Sigma_{\mathrm{i}}$ and $S' \models'_{SiS} \phi$. Then$(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \models_{SiS})$ and $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \models'_{SiS})$ result the exact same Fitting twisted enough-certainty semantics.*

Proof. The proof of this is rather obvious from the definitions. The idea is that if for a significant set S, it is true that for all significant $S' \subseteq S$ there is a significant $S'' \subseteq S'$ such that $V, S'' \models_{\mathrm{SiS}} \phi$ with ϕ atomic, then there is a significant $S''' \subseteq S''$ such that $V, S''' \models'_{\mathrm{SiS}} \phi$ and vice versa by the conditions of the proposition. Then for compound formulas induction can be applied with the same idea. ◀

As a consequence of this, it does not matter if in Example 20 we require that $\mathcal{A}$ succeeds on the S'' sets completely or with a negligible error because they result the same exact definition. Moreover, if we apply the Fitting twist to $\models_{\mathrm{KAC}}$ and $\models_{\mathrm{KEC}}$ (as special instances of $\models_{\mathrm{SiS}}$), the resulting semantics is the same.

The theorem below shows that the covering enough-certainty semantics in certain cases is just a special case of the Fitting twist.

▶ **Proposition 23.** *Let $(\mathcal{D}, W, \Sigma, \Sigma_{\mathrm{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ be a possible world model with significant events. If $\mathcal{D}$ only has countably many elements, and if $\{w : w \models \phi\} \in \Sigma$ for all $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$, and if Σ_{i} is a σ-subalgebra then covering enough-certainty semantics is a special case of Fitting twisted enough-certainty semantics. In other words, when for the Kripke semantics $\models_{KSiS}$, the special case $\models_{KAC}$ is taken, then the Fitting twisted semantics $\models_{FEC}$ is the same as $\models_{CEC}$.*

Outline of the proof: It is shown first that with the conditions of the theorem, for any formula ϕ, and valuation V, there is a maximal $S \in \Sigma_s$ such that $V, S \models_{\text{CEC}} \phi$. With this idea, the proof is rather straightforward induction on the size of the formulas, and it is important that countable unions of insignificant sets are still insignificant with the conditions.

4 Soundness Of Classical Logic With Respect To Enough-Certainty Semantics

In this section we relate our semantics introduced in Section 2 and 3 to the syntactic deducibility/provability notion of classical (first order) predicate logic, in terms of soundness. Then we reconsider the examples we saw earlier to show the usage of this soundness theorem. First-order logic is sound and complete with respect to the standard first-order semantics (that is, Tarskian semantics, when the interpretation of connectives and quantifiers are defined the usual way), which has been well known since Gödel (1930). It has also been explored from time to time that soundness of first-order logic can be maintained for various ranges of semantics which are wider than the standard Tarskian semantics. For example, Rasiowa and Sikorski [13] introduced Boolean valued models for which first order logic is sound. The Boolean-valued classical semantics was used later by Scott-Solovay [14] for a semantic framework of Cohen's forcing model-construction in set theory; we can view this as an example of the usefulness of such extended classical semantics. As we shall discuss later in detail, Fitting introduced in [7] a possible world semantics for classical logic, which corresponds to "necessity-possibility" (or box-diamond) interpretation in terms of S4-modal logical syntactic interpretation. Fitting's interpretation uses double modalities (we can say "twisted" interpretation), which is effectively constructing from standard classical semantics a new semantics that is still classical in the sense that first-order deduction rules are sound, but not standard as some connectives and quantifications are not interpreted as usual. Note, when we say first-order deduction rules are sound, we mean *any* first-order deduction system, including natural deduction, that is sound for standard interpretations.

A different way to obtain from standard classical semantics a new classical semantics via S4 is to combine the Gödel's modal embedding of intuitionistic logic in S4 [9] and the Gödel-Kolmogorov embedding of classical logic into intuitionistic logic [10, 12]. These and various other embedding theorems are ways of constructing new classical semantics from some base classical semantics.

In Section 4.1, we use Fitting's theorem to show that soundness holds for our Fitting-twisted semantics. One natural way to extend the classical semantics is to introduce a new semantic notion. In our case, we are motivated to introduce "enough-certainty" and its variations. In Section 2 and 3, we reached such a new semantics and here we confirm that the new semantics is still classical semantics. We then finish this section by illustrating the use of this new semantics on the examples we have introduced.

4.1 Soundness Theorems

In this section we discuss the relationship of our Fitting-twisted semantics and Fitting's embedding of first-order logic into first-order S4.

In 1970, Melvin Fitting published a paper [7] that is about embedding first order logic into first-order S4. He does this in the following way. For any first-order formula θ, consider the transformation $\theta \to \theta^*$, where θ^* is a formula of S4 (with Barcan formulas), and is defined recursively as follows:

- For any atomic formula θ, let $\theta^* \equiv \Box\Diamond\theta$.
- $(\neg\theta)^* \equiv \Box\neg\theta^*$

- $(\theta_1 \to \theta_2)^* \equiv \Box(\theta_1^* \to \theta_2^*)$
- $(\theta_1 \wedge \theta_2)^* \equiv (\theta_1^* \wedge \theta_2^*)$
- $(\theta_1 \vee \theta_2)^* \equiv \Box\Diamond(\theta_1^* \vee \theta_2^*)$
- $(\forall x\theta)^* \equiv \forall x\theta^*$
- $(\exists x\theta)^* \equiv \Box\Diamond\exists x\theta^*$

Fitting in [7] put $\Box\Diamond$ everywhere and noted that it is redundant in front of the conjunction. It is also easy to check that if the Barcan formula and its converse ($\forall x\Box\theta \leftrightarrow \Box\forall x\theta$) are assumed (that is, when the domain does not change from possible world to possible world in the Kripke structure), then $\Box\Diamond$ is also redundant in front of the universal quantification (as $\theta^* \leftrightarrow \Box\Diamond\theta^*$ holds in our definitions for all θ). For us it is sufficient to only consider a single domain for each possible world, so we assume the Barcan formula and its converse. Fitting's original theorem works for varying domains as well. Fitting's theorem says the following:

- **Fitting's Embedding.** Any formula θ is derivable in first-order logic if and only if θ^* it is derivable in S4 with the Barcan formulas. (Without the Barcan formulas, the theorem still holds, but $(\forall x\theta)^* \equiv \Box\Diamond\forall x\theta^*$ has to be written above).

Fitting's result is closely related to forcing, and a satisfaction relation in S4, $\Gamma \models \Box\Diamond\theta$ is analogous to "θ is weakly forced". While forcing is used in logic to find models of certain formulas, as we showed in the previous section, the above construction naturally arises in the field of computer security when formalizing and deriving properties are based on computational complexity. Let $\vdash_{FOL}$ denote the usual first-order deduction.

▶ **Theorem 24.** *For any first-order formula ϕ, we have that $\vdash_{FOL} \phi$ if and only if it is satisfied by all Fitting-twisted enough-certainty models.*

Proof. The if part follows from the completeness of first-order logic: as a first-order model is a trivial Fitting twisted enough-certainty model, a formula that is satisfied by all Fitting twisted enough-certainty models is also satisfied by all first-order models.

The only if part follows from Fitting's theorem for his embedding. The satisfaction of $\models_{\text{FEC}}$ is the combination of Fitting embedding with $\models_{\text{KSiS}}$: if for a $\phi \in \Phi_{(\mathfrak{f},\mathfrak{p})}$, ϕ^* denotes the Fitting interpretation of ϕ, then it is immediate from the definitions that

$$V, S \models_{\text{FEC}} \phi \Leftrightarrow V, S \models_{\text{KSiS}} \phi^*$$

Since $\models_{\text{KSiS}}$ is a special Kripke semantics and hence S4 deduction rules are sound, and since by Fitting's theorem a formula of the form ϕ^* can be deduced in $S4$ if and only if it can be deduced in first-order logic, we have that if ϕ^* can be deduced in first-order logic then $V, S \models_{\text{KSiS}} \phi^*$ and hence $V, S \models_{\text{FEC}} \phi$ holds. ◀

▶ **Corollary 25.** *Let $(\mathcal{D}, W, \Sigma, \Sigma_{\text{s}}, \mathfrak{f}, \mathfrak{p}, \models)$ be a possible world model with significant events. If $\mathcal{D}$ is countable, if $\{w \in W : w \models \phi\}$ is measurable for all $\phi \in A_{(\mathfrak{f},\mathfrak{p})}$, and if Σ_{i} is a σ-subalgebra, then although the semantics of compound formulas with respect to $\models_{CEC}$ is not defined the usual Tarski way, first order deduction rules are valid with respect to $\models_{CEC}$.*

Proof. This is a direct consequence of Theorem 24 and Proposition 23 ◀

4.2 Applications to the Examples

In this section we come back to the examples that we introduced earlier, and see how Fitting's embedding can be applied there and use classical logic for the deduction of desirable properties.

▶ **Example 26.** In Example 14, we looked at how $\to$ becomes interpreted as conditional implication under $\models_{\mathrm{CEC}}$. We also saw in Corollary 25 that when $\mathcal{D}$ is countable, then the first-order deduction rules are sound for $\models_{\mathrm{CEC}}$. However, even when $\mathcal{D}$ is not countable, the Fitting-twisted semantics obtained from $\models_{\mathrm{AC}}$, as a special case of $\models_{\mathrm{FEC}}$ also gives conditional implication for formulas without quantifiers: When $\models_{\mathrm{FEC}}$ is a Fitting-twisted $\models_{\mathrm{AC}}$, then

$$V, S \models_{\mathrm{FEC}} \phi \to \psi \quad \Leftrightarrow \quad [\neg\psi]_V \wedge [\phi]_V \in \Sigma_{\mathrm{i}}$$

as long as $[\neg\psi]_V, [\phi]_V \in \Sigma$.

In case of a parametrized probability space $(\Omega, \Sigma, \mathcal{P})$, it is easy check that for $S \in \Sigma_{\mathrm{s}}$,

$$V, S \models_{\mathrm{FEC}} \phi \to \psi \quad \Leftrightarrow \quad \Big(1 - P_i\big(\ [\psi]_V \ \big|\ [\phi]_V\ \big)\Big)_{i\in\mathbb{N}} \in \mathcal{C}$$

As a result, even if $\mathcal{D}$ is not countable, it is possible to use first-order logic for conditional implication, when we are only interested in this implication on significants sets.

▶ **Example 27.** As it was worked out in [3, 1, 4], when a security protocol is executed, there are random inputs, coins are tossed by both the attacker and the honest agents. Random nonces are generated, encryptions usually use random inputs, the attacker is also allowed to toss coins. The agents' and the attacker's actions may be different for the various random inputs. A property that may hold for some of the random inputs may fail for others. The security property of a protocol takes the form $\phi \to \psi$, which, in the Fitting-twisted semantics means, as we mentioned in the Remark after Definition 21, that if ϕ is satisfied on a non-negligible part of a protocol execution, then ψ is also satisfied on that non-negligible part (except maybe for negligible probability). That is, whenever the properties expressed by ϕ are satisfied on a part of the execution, the properties expressed by ψ are also satisfied. On other parts, were ϕ is not satisfied, ψ is not required to be satisfied either.

For example ϕ could be expressing that a random secret (nonce) was communicated between two honest agents and ψ expressing that this nonce cannot be computed by the attacker. In this case ψ has the form $t_1, \ldots, t_n \not\rhd N$ (see Example 20) where N is the nonce, $t_1, \ldots, t_n$ are the messages the attacker has seen. Then such a security property is deduced from the axioms on the predicate $\rhd$. Some axioms are general axioms, such as $t_1, \ldots, t_n \rhd t \to t_1, \ldots, t_n, u \rhd t$ (expressing that if the attacker can compute t without u, then it can also compute t with u), while some express the security of the cryptographic primitives such as (roughly) $t_1, \ldots, t_n, \{u\}^r_k \rhd t \to t_1, \ldots, t_n \rhd t$ (expressing that the encryption $\{u\}^r_k$ cannot help computing t).

Because of the soundness theorem Theorem 24, first order deduction rules can be used to deduce the security property from the axioms, and as a result, complexity theoretic guarantees will be obtained. Verifying complexity-theoretic properties of security protocols is a difficult problem and although various research groups have put large efforts to create automated tools for this purpose, there has not been real breakthrough in this area. The aim of this technique is to allow the use of a fragment of first-order logic, the automation of which is well researched and hence open the possibility of fully automated proofs with complexity-theoretic guarantees.

If the semantics of $\to$ were the usual Tarski semantics, then it would mean that if the premise is satisfied over the entire execution space then the conclusion is also satisfied over the entire space. This however is not appropriate because we need to be able to reason about parts of the execution space. The solution of the authors of [6] was to introduce two kinds of implication: one that has the usual Tarski interpretation and one $\to'$ that has a conditional interpretation coming from ϵ-semantics. However, their conditional implication has some

undesirable non-classical properties such as $(\phi \rightarrow' \psi) \not\rightarrow (\phi \wedge \phi' \rightarrow' \psi)$, which we would only need if we cared what happened on negligible sets. And why use two implications when one is sufficient?

5 Conclusion

In this work we have introduced a semantics for "enough-certainty" that allows us to argue with first-order logic about satisfaction of properties on significant sets of possible worlds while ignoring what happens on insignificant sets of possible worlds. We were motivated by an application to computer security and complexity theory, we showed how this non-Tarskian semantics emerged entirely naturally in the context of computer security, and presented a more general framework shifting from non-negligible sets of complexity theory to a notion of significant sets. The trick that ensures the soundness of first-order deduction rules is to combine an S4 Kripke-semantics based on the significant sets as possible worlds with Fitting's embedding of first-order logic in first-order S4. We also showed how the Fitting twist turns material implication into conditional implication.

References

1 G. Bana, P. Adão, and H. Sakurada. Computationally Comlete Symbolic Attacker in Action. In *Proceedings of the 32nd International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, LIPIcs, pages 546–560. Schloss Dagstuhl, 2012.

2 G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. Available at IACR ePrint Archive, Report 2012/019.

3 G. Bana and H. Comon-Lundh. Towards unconditional soundness: Computationally complete symbolic attacker. In *Proceedings of the 1st International Conference on Principals of Security and Trust (POST'12)*, LNCS, pages 189–208. Springer, 2012.

4 G. Bana, K. Hasebe, and M. Okada. Computationally complete symbolic attacker and key exchange. In *Proceedings of the 20th ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*, pages 1231–1246. ACM, 2013.

5 P. J. Cohen. *Set theory and the continuum hypothis*. W. A. Benjamin, Inc., New York, 1966.

6 A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 321–334. IEEE, 2006.

7 M. Fitting. An embedding of classical logic in s4. *The Journal of Symbolic Logic*, 35(4):529–534, 1970.

8 N. Friedman, J. Y. Halpern, and D. Koller. First-order conditional logic for default reasoning revisited. *ACM Transactions on Computational Logic*, 1(2):175–207, 2000.

9 K. Gödel. Eine interpretation des intuitionistischen aussagenkalküls. *Ergebnisse eines Mathematischen Kolloquiums*, 4:39–40, 1933.

10 K. Gödel. Zur intuitionistischen arithmetik und zahlentheorie. *Ergebnisse eines Mathematischen Kolloquiums*, 4:34–38, 1933.

11 M. Goldszmidt, P. Morris, and J. Pearl. A maximum entropy approach to nonmonotonic reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):220–232, March 1993.

12 A. N. Kolmogorov. O principe tertium non datur (russian). *Matematiceskij Sbornik*, 32:646–667, 1925.

13 H. Rasiowa and R. Sikorski. *The Mathematics of Metamathematics*. Polish Scientific Publishers, 1963.

14 D. Scott and R. Solovay. Boolean-valued models of set theory. *unpublished and circulated*, 1967.

15 R. M. Smullyan and M. Fitting. *Set Theory and the Continuum Problem*. Oxford University Press, 1996. revised addition by Dover, 2006.

16 J. Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*. Cambridge University Press, 2007.

Counting in Team Semantics

Erich Grädel[1] and Stefan Hegselmann[2]

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

2 **Mathematical Foundations of Computer Science, RWTH Aachen University, Aachen, Germany**
hegselmann@logic.rwth-aachen.de

Abstract

We explore several counting constructs for logics with team semantics. Counting is an important task in numerous applications, but with a somewhat delicate relationship to logic. Team semantics on the other side is the mathematical basis of modern logics of dependence and independence, in which formulae are evaluated not for a single assignment of values to variables, but for a set of such assignments. It is therefore interesting to ask what kind of counting constructs are adequate in this context, and how such constructs influence the expressive power, and the model-theoretic and algorithmic properties of logics with team semantics. Due to the second-order features of team semantics there is a rich variety of potential counting constructs. Here we study variations of two main ideas: *forking atoms* and *counting quantifiers*.

Forking counts how many different values for a tuple $\overline{w}$ occur in assignments with coinciding values for $\overline{v}$. We call this the *forking degree* of $\overline{v}$ with respect to $\overline{w}$. Forking is powerful enough to capture many of the previously studied atomic dependency properties. In particular we exhibit logics with forking atoms that have, respectively, precisely the power of dependence logic and independence logic.

Our second approach uses counting quantifiers $\exists^{\geq\mu}$ of a similar kind as used in logics with Tarski semantics. The difference is that these quantifiers are now applied to teams of assignments that may give different values to μ. We show that, on finite structures, there is an intimate connection between inclusion logic with counting quantifiers and FPC, fixed-point logic with counting, which is a logic of fundamental importance for descriptive complexity theory. For sentences, the two logics have the same expressive power. Our analysis is based on a new variant of model-checking games, called threshold safety games, on a trap condition for such games, and on game interpretations.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases logics with counting, team semantics, fixed-point logic with counting

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.35

1 Introduction

> Ravens, so we read, can only count up to seven. They can't tell the difference between two numbers greater than or equal to eight. First-order logic is much the same as ravens, except that the cutoff point is rather higher: it's ω instead of 8. Wilfrid Hodges

Logic and counting. Counting the number of elements satisfying a certain property is a basic task of fundamental importance that arises in many applications. While this is computationally easy (if the underlying property is decidable in a simple way) it is problematic

© Erich Grädel and Stefan Hegselmann;
licensed under Creative Commons License CC-BY

for many classical logical systems. The quotation by Hodges [9] only shows the tip of the iceberg. If we do not look at first-order theories but at single sentences, then the cutoff point (depending on vocabulary and quantifier-rank) may be much lower, and the inability to count persists for logics that are much stronger than first-order logic, such as for instance fixed-point logics. In particular, the computationally trivial query of determining whether a given finite structure has an even or odd number of elements is not definable by any formula from first-order logic (FO), fixed-point logic (LFP), or even from $L^{\omega}_{\infty\omega}$, the infinitary logic with a bounded number of variables.

In finite model theory, a lot of attention has therefore been given to logics that incorporate counting in some way or another, for instance by counting quantifiers or counting terms, or by generalized quantifiers for cardinality comparison such as Rescher or Härtig quantifiers. The simplest way to add counting is in terms of quantifiers of form $\exists^{\geq i} x\varphi$ for fixed $i \in \mathbb{N}$, saying that there exist at least i distinct values for x satisfying φ. While such counting quantifiers do not add anything to the expressive power of full first-order logic or fixed-point logic, they are relevant for the study of logics with a bounded number of variables, such as C^k or $C^k_{\infty\omega}$. A more powerful variant of counting is obtained with counting terms of form $\#\varphi(x)$ or counting quantifiers $\exists^{\geq\mu}$ or $\exists^{\leq\mu}$ where μ is a variable, and the values assumed by μ or by counting terms are natural numbers (that are kept separate from the elements of the structure). Thus, logics with this kind of counting are evaluated on two-sorted structures $\mathfrak{A}^* = \mathfrak{A} \cup (\omega, <, +, \cdot, 0, e)$, i.e., finite structures expanded by a disjoint ordered numeric sort (here the natural numbers with arithmetic but there are other possible choices). The most important counting logic, at least in finite model theory, is fixed-point logic with counting (FPC). It has first been proposed, somewhat informally, by Immerman, later a more formal definition based on two-sorted structures, counting terms, and inflationary fixed-points of relations ranging over both sorts has been adopted. Meanwhile FPC has become the logic of reference in the search for a logic for Ptime. Although it has been known since the 1990s, by a fundamental construction due to Cai, Fürer, and Immerman, that FPC fails to express all polynomial-time queries, it comes rather close to being a logic for polynomial time. It is strong enough to express most of the fundamental algorithmic techniques leading to polynomial-time procedures and it captures Ptime on many interesting classes of finite structures, including trees, planar graphs, structures of bounded tree width, and actually all classes of graphs with an excluded minor. For a survey on FPC, and for references, see [1].

Logics with team semantics. In this paper, we study counting constructs for team semantics. The idea of team semantics goes back to a paper by Hodges [10] where he provided a model-theoretic, compositional, semantics for the independence-friendly logic IF, as an alternative to the semantics based on games of imperfect information or on Skolem functions. In team semantics a formula $\varphi(x_1, \dots x_k)$ is evaluated, on a given structure $\mathfrak{A}$, not for a single assignment $s : \{x_1, \dots, x_k\} \to A$ but for a *set* of such assignments, and, following [15], a set of assignments with a common (finite) domain of variables is called a team. Personally we find that the invention of team semantics by Wilfrid Hodges is really a major innovation in logic. Combined with Väänänen's proposal [15] to treat dependencies as atomic statements, and not as annotations of quantifiers, it has lead to a genuinely new area in logic, with an interdisciplinary motivation of providing logical systems for reasoning about the fundamental notions of dependence and independence that permeate many scientific disciplines. Methods from several areas of computer science, including finite model theory, database theory, and the algorithmic analysis of games have turned out as highly relevant for this area.

Notice that statements about dependence or independence, such as *"z is functionally dependent on x and y"* or *"x and y are independent"* do not make much sense for a single

assignment to the variables, but require larger amounts of data, as given by a table or relation, or by a team of assignments. Team semantics is therefore the natural mathematical basis for the modern logics of dependence and independence in which dependency or independency statements are basic atomic building blocks, similar to equality statements. The best studied logic with team semantics is dependence logic, which extends first-order logic by dependency atoms of form $=(\overline{x},\overline{y})$, saying that the variables $\overline{y}$ are functionally dependent on (i.e. completely determined by) the variables $\overline{x}$, but there are many other atomic dependence properties that give rise to interesting logics based on team semantics. In [7] we have discussed the notion of independence (which is a much more delicate but also more powerful notion than dependence) and introduced independence logics, and Galliani [4] and Engström [3] have studied several logics with team properties based on notions originating in database dependency theory. Of particular interest for us is *inclusion logic* FO($\subseteq$) which extends first-order logic by atomic inclusion dependencies ($\overline{x} \subseteq \overline{y}$), which are true in a team X if every value for $\overline{x}$ in X also occurs as a value for $\overline{y}$ in X. There is also a dual notion, exclusion logic, based on exclusion statements ($\overline{x} \mid \overline{y}$), saying that $\overline{x}$ and $\overline{y}$ have disjoint sets of values in the team X. Exclusion logic has turned out to be equivalent to dependence logic [4].

Expressive power of logics with team semantics. If we study the expressive power of such logics, for instance by comparison to classical logics, we have to keep in mind the different nature of team semantics and Tarski semantics. For a formula with team semantics, we write $\mathfrak{A} \models_X \varphi$ to denote that φ is true in the structure $\mathfrak{A}$ for the team X, and for classical Tarski semantics we write $\mathfrak{A} \models_s \varphi$ to denote that φ is true in $\mathfrak{A}$ for the assignment s. A direct comparison is possible in the case of sentences. For any sentence ψ from a logic with team semantics, we write $\mathfrak{A} \models \psi$ if $\mathfrak{A} \models_{\{\emptyset\}} \psi$, i.e. if ψ is true for the team $X = \{\emptyset\}$ that consists just of the empty assignment[1]. For formulae with free variables the translation from a logic with team semantics into one with Tarski semantics requires that we represent the team in some way. The standard way to do this is by identifying a team X of assignments $s : \{x_1, \dots x_k\} \to A$ with the relation $\{s(\overline{x}) \in A^k : s \in X\} \subseteq A^k$ which, by slight abuse of notation, we also denote by X. One then translates formulae $\varphi(x_1, \dots, x_k)$ of vocabulary τ into *sentences* φ^* of the expanded vocabulary $\tau \cup \{X\}$ such that for every structure $\mathfrak{A}$ and every team X we have that

$$\mathfrak{A} \models_X \varphi(x_1, \dots, x_k) \iff (\mathfrak{A}, X) \models \varphi^*.$$

In all logics with team semantics that extend first-order logic (or a fragment thereof) by atomic dependency statements that are themselves first-order definable, and which do not make use of additional connectives beyond $\wedge, \vee$ and atomic negation, such a translation will always produce sentences in (a fragment of) existential second-order logic, denoted Σ^1_1. Understanding the expressive power of a logic L with team semantics thus means to identify the fragment of Σ^1_1 to which L is equivalent in the sense just described. The following is known in this context:

1. Dependence logic is equivalent to the fragment of all Σ^1_1-sentences $\psi(X)$ in which the predicate X describing the team appears only negatively [12].
2. Independence logic and inclusion-exclusion logic are equivalent with full Σ^1_1 (and thus can describe all NP-properties of teams) [4].

[1] Notice that we cannot replace this by the empty team $X = \emptyset$. The common logics with teams semantics have the empty team property which means that the empty team satisfies all formulae.

3. The extension of FO by inclusion and exclusion atoms of single variables only (not tuples of variables) is equivalent to monadic Σ^1_1 [14].
4. First-order logic without any dependence atoms has the so-called *flatness property*: $\mathfrak{A} \models_X \varphi \iff \mathfrak{A} \models_{\{s\}} \varphi$ for all $s \in X$. It thus corresponds to a very small fragment of Σ^1_1, namely FO-sentences of form $\forall \overline{x}(X\overline{x} \to \varphi(\overline{x}))$ where $\varphi(\overline{x})$ does not contain X.

For our study the most interesting result of this kind concerns the relationship of inclusion logic FO($\subseteq$) with posGFP, the fragment of LFP that uses only (non-negated) greatest fixed points. Since a greatest fixed-point formula $[\mathbf{gfp}\, R\overline{x}\,.\,\psi(R,\overline{x})](\overline{y})$ readily translates into $(\exists R)((\forall \overline{x}(R\overline{x} \to \psi(R,\overline{x})) \wedge R\overline{y}))$, posGFP can be viewed as a fragment of Σ^1_1. Galliani and Hella [5] established translations between inclusion logic and posGFP that extend the list of equivalences between logics with team semantics and fragments of Σ^1_1 by

5. Inclusion logic is equivalent to the set of sentences of form $\forall \overline{x}(X\overline{x} \to \psi(X,\overline{x}))$, where $\psi(X,\overline{x})$ is a formula in posGFP in which X occurs only positively. In particular, on any finite structure, the maximal team satisfying a formula $\varphi(\overline{x})$ of inclusion logic coincides with the greatest fixed point of (the operator defined by) the posGFP-formula $\psi(X,\overline{x})$.

A different proof for this result, based on safety games and game interpretations, has been presented in [6]. Notice that for sentences, inclusion logic and posGFP have the same expressive power. It is known that, on finite structures, the full logic LFP collapses to its posGFP-fragment [11]. Hence every property of finite structures that is LFP-definable is also definable in inclusion logic, and vice versa. It follows by the Immerman-Vardi-Theorem that, on *ordered* finite structures, inclusion logic captures polynomial time.

Counting constructs for team semantics. The relevance of logics with counting for finite model theory, and the known connections between logics with team semantics (such as inclusion logic) and logics that are important in finite model theory (such as LFP), raise the question of counting constructs for logics with team semantics, and how these enhance the expressive power of such logics. The second-order nature of team semantics in fact leads to a rich variety of potential counting constructs for logics with team semantics, because there are several different objects that one may wish to count.

We illustrate this variability with the well-known *majority quantifier* M. In Tarski semantics a formula $My\varphi(\overline{x},y)$ expresses, on a finite structure $\mathfrak{A}$ and an assignment $s : \overline{x} \mapsto \overline{a}$, that $\mathfrak{A} \models \varphi(\overline{a},b)$ for at least half of the possible values b for y. In particular, $MyExy$ is true for a graph $G = (V,E)$ and an assignment $s : x \mapsto v$ if v is adjacent to at least half of the nodes in G. There are several possibilities to define the team semantics of $MyExy$. A team X of assignments to x describes a subset $U := X[x] \subseteq V$. We may define that $G \models_X MyExy$ if *every node in* U is adjacent to at least half of the nodes in V. We shall see that this corresponds to defining majority via *counting quantifiers*. But we could also define that $G \models_X MyExy$ if U *as a set* is adjacent to at least half of the nodes in G, or, that at least half of the nodes in G are adjacent to *all* elements of U. These two possibilities are related to a different kind of counting that we call *forking*.

In all three cases, this amounts to a general definition of the team semantics of the majority quantifier, saying that $\mathfrak{A} \models_X My\varphi(\overline{v},y)$ if, and only if, $\mathfrak{A} \models_{X[y\mapsto F]} \varphi$ for some appropriate function $F : X \to \mathcal{P}(A)$, but the requirements for F are different. In the first case we demand that $|F(s)| \geq |A|/2$ for all $s \in X$, in the second case that $|\bigcup_{s\in X} F(s)| \geq |A|/2$, and in the third case that $F(s) = B$ for all $s \in X$ and some fixed set $B \subseteq A$ with $|B| \geq |A|/2$.

In the sequel, we shall focus on variations of two main ideas: *forking* and *counting quantifiers*. Forking counts, for variables $\overline{v}$ and $\overline{w}$, how many different values for $\overline{w}$ occur in

assignments with coinciding values for $\overline{v}$. We call this the *forking degree* of $\overline{v}$ with respect to $\overline{w}$. Notice that a forking degree of one is equivalent to functional dependence, so forking can be seen as a generalization of dependence. We shall introduce a counting mechanism for the forking degree by means of *forking atoms* over teams. We examine closure properties and expressive power of different extensions of first-order logic with forking atoms.

Our second approach uses counting quantifiers $\exists^{\geq\mu}$ of a similar kind as used in, say, fixed-point logic with counting. The difference is that these quantifiers are now applied to teams of assignments that may give different values to μ. A counting quantifier $\exists^{\geq\mu}x$ thus requires a witness that extends a given team X by adding to each $s \in X$ a set of at least $s(\mu)$ many distinct values for x. A formal definition will be given in Sect. 4. It is not hard to see that in logics that have the full power of Σ^1_1 (or come close to it) this form of counting is definable, even without a second numeric sort. Thus counting quantifiers are interesting mainly for logics with team semantics that are strictly weaker than Σ^1_1, in particular for inclusion logic. We shall analyse the power of logics with counting by an appropriate variant of games that we call threshold safety games. Our main result will indeed give an equivalence, in the sense described above, between inclusion logic with counting, $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$, and fixed-point logic with counting, FPC.

2 Preliminaries

Two-sorted structures. With every finite relational structure $\mathfrak{A}$, we associate the two-sorted structure $\mathfrak{A}^* = \mathfrak{A} \cup (\omega, <, +, \cdot, 0, e)$, where A (the universe of $\mathfrak{A}$) and ω (the set of natural numbers) are assumed to be disjoint. Let $A^* := A \cup \omega$. We call $\mathfrak{A}$ the point sort, and $(\omega, <, +, \cdot, 0, e)$ the numeric sort. The numeric constant e stands for $|A|$, the cardinality of the point sort. All variables are typed: we use Latin letters $x, y, z, \ldots$ for variables over the point sort, and Greek letters $\mu, \nu, \lambda, \ldots$ for variables over the numeric sort. Whenever the sort of a variable is irrelevant and not clear from context, we shall use the variable symbols v or w. For some applications, it is relevant to generalize such structures to expansions $(\mathfrak{A}^*, F)$ with a set F of functions $f : A^k \to \omega$.

Quantifiers over the numeric sort must be bounded, of the form $(\forall\mu < t)$ or $(\exists\mu < t)$, where t is a closed numeric term (but for ease of notation we shall suppress this in cases where the bounds are irrelevant or clear from context). Such bounds could be avoided altogether, if one would use a finite numeric sort, such as $(\{0, \ldots, |A|\}, <, 0, e)$ instead of the natural numbers. Both approaches have their advantages, but here we use an infinite numeric sort to make the counting of tuples of elements simpler. We write $\omega_{<t}$ for the set of natural numbers smaller than (the value of) t.

Team semantics. From the start, we adapt the notion of teams from [15] to two-sorted structures $\mathfrak{A}^*$. For a set $\mathcal{V}$ of (typed) variables, an *assignment* into $\mathfrak{A}^*$ is a map $s : \mathcal{V} \to A^*$ that respects the types of the variables. Given such an assignment s, a tuple $\overline{v} = (v_1, \ldots, v_k)$ of distinct variables, and elements $c_1, \ldots, c_k \in A^*$ (with types corresponding to those of $\overline{v}$) we write $s[\overline{v} \mapsto \overline{c}]$ for the assignment with domain $\mathcal{V} \cup \{v_1, \ldots, v_k\}$ that updates s by mapping v_i to c_i, for $i = 1, \ldots, k$.

▶ **Definition 1.** A *team* is a set of assignments with the same domain into a structure $\mathfrak{A}^*$. For a tuple of variables $\bar{v} = (v_1, ..., v_n)$, let $s(\bar{v}) := (s(v_1), ..., s(v_n))$ and let $X[\bar{v}] := \{s(\bar{v}) : s \in X\}$ denote the set of values for $\overline{v}$ in X. For a team X, a k-tuple $\overline{v}$, and a function $F : X \to \mathcal{P}((A^*)^k)$, we write $X[\overline{v} \mapsto F]$ for the set of all assignments $s[\overline{v} \mapsto \overline{c}]$ with $s \in X$ and $\overline{c} \in F(s)$.

Team semantics, for a logic L, defines whether a formula $\psi \in L$ is satisfied by a team X in a structure $\mathfrak{A}^*$, written $\mathfrak{A}^* \models_X \psi$. We always assume formulae to be in negation normal form and require that X is correctly typed, with a domain that contains all free variables of ψ. Following the approach by Väänänen [15], modern logics of dependence and independence are based on atomic dependency properties of teams. The most important ones, for this paper, are

Dependence: A dependence atom has the form $=(\overline{v}, \overline{w})$. It is true in a team X if, and only if, $s(\overline{w}) = s'(\overline{w})$ for all assignments $s, s' \in X$ with $s(\overline{v}) = s'(\overline{v})$.

Constancy: A special case of a dependence atom is the constancy atom $=(v)$, saying that v assumes only one value in X.

Independence: A (pure) independence atom[2] has the form $\overline{v} \perp \overline{w}$. It holds in a team X if, and only if, for all $s, s' \in X$ there is a third assignment $s'' \in X$ such that $s''(\overline{v}) = s(\overline{v})$ and $s''(\overline{w}) = s'(\overline{w})$.

Inclusion: An inclusion atom has the form $(\overline{v} \subseteq \overline{w})$, for tuples $\overline{v}, \overline{w}$ of the same length and type. It is true in a team X if, and only if, $X[\overline{v}] \subseteq X[\overline{w}]$, i.e. if every value that occurs for $\overline{v}$ in X also occurs as a value for $\overline{w}$.

In the logics with team semantics that we consider, atomic dependency properties are used only positively. Thus, negation is applied only to first-order atoms. A first-order literal $\alpha(\overline{v})$ is defined to be true for a team X in the structure $\mathfrak{A}^*$ if it is true, in the sense of classical Tarski semantics, for all individual assignments $s \in X$, i.e.

$$\mathfrak{A}^* \models_X \alpha(\overline{v}) \iff \mathfrak{A}^* \models_s \alpha(\overline{v}) \text{ for all } s \in X.$$

The common logics of dependence and independence extend first-order literals and atomic dependency properties by the usual first-order connectives and quantifiers $\vee, \wedge, \exists, \forall$ to obtain full-fledged logics for reasoning about dependency properties. The semantic rules for these, in the context of two-sorted structures, are the following:

- $\mathfrak{A}^* \models_X (\varphi \wedge \vartheta)$ if, and only if, $\mathfrak{A}^* \models_X \varphi$ and $\mathfrak{A}^* \models_X \vartheta$.
- $\mathfrak{A}^* \models_X (\varphi \vee \vartheta)$ if, and only if, there exist teams Y, Z with $X = Y \cup Z$ such that $\mathfrak{A}^* \models_Y \varphi$ and $\mathfrak{A}^* \models_Z \vartheta$.
- $\mathfrak{A}^* \models_X \exists y \varphi$ if, and only if, there is a map $F : X \to (\mathcal{P}(A) \setminus \{\emptyset\})$ such that $\mathfrak{A}^* \models_{X[y \mapsto F]} \varphi$.
- $\mathfrak{A}^* \models_X \forall y \varphi$ if, and only if, $\mathfrak{A}^* \models_{X[y \mapsto A]} \varphi$ where $X[y \mapsto A] := \{s[y \mapsto a] : s \in X, a \in A\}$.
- $\mathfrak{A}^* \models_X (\exists \mu < t)\varphi$ if, and only if, there is a map $F : X \to (\mathcal{P}(\omega_{<t}) \setminus \{\emptyset\})$ such that $\mathfrak{A} \models_{X[\mu \mapsto F]} \varphi$.
- $\mathfrak{A}^* \models_X (\forall \mu < t)\varphi$ if, and only if, $\mathfrak{A} \models_{X[\mu \mapsto \omega_{<t}]} \varphi$ where $X[\mu \mapsto \omega_{<t}] := \{s[\mu \mapsto m] : s \in X, m < t\}$.

First-order logic extended with dependence atoms, constancy atoms, inclusion atoms, and independence atoms, are, respectively called dependence logic FO(dep), constancy logic FO(const), independence logic FO(indep), and inclusion logic FO($\subseteq$).

The locality principle. In our definitions we shall always make sure that the *locality principle* holds, saying that the meaning of a formula can only depend on the variables actually occurring in it. More precisely, if $Y = X \restriction \mathrm{free}(\psi)$ is the restriction of the team X to the free variables of ψ then $\mathfrak{A} \models_X \psi$ if, and only if, $\mathfrak{A} \models_Y \psi$. As obvious as it seems, the

[2] There is also a more general variant of independence atoms, but it is known that these are expressible by means of pure independence atoms.

locality principle is not trivial, and it it easy to violate it by choosing 'wrong' definitions. Consider for instance the seemingly more natural semantics (called strict semantics) for existential quantifiers, that defines a formula $\exists y\varphi$ to hold for a team X if, and only if, there exists a function $F : X \to A$ such that φ holds for the team of all assignments $s[y \mapsto F(s)]$ with $s \in X$. While such a choice of a single witness, rather than a non-empty set of witnesses, for an existentially quantified variable would make no difference for downwards closed logics such as pure first-order logic or dependence logic, it would lead to a violation of the locality principle when combined with, say, inclusion atoms. Indeed, even the simple formula $\exists x(y \subseteq x \wedge z \subseteq x)$, which trivially holds for all teams under the semantics given above, may depend under strict semantics on the presence of additional variables in the domain of the team.

More importantly for us, the locality principle may be violated by certain forms of counting, such as counting quantifiers of form $\exists^{\leq \mu} x$. Thus the locality principle provides a kind of sanity check for the definition of logical operators in team semantics, and we shall make sure that all our proposals pass this check.

Closure properties. One way to study the properties and power of logics with team semantics is based on the operations on teams that preserve the truth of their formulae. For instance, an important property of dependence logic is *downwards closure*: whenever a formula of FO(dep) is true for a team X, then it is also true for all subteams $Y \subseteq X$. Inclusion logic, on the other side, is not downwards closed, but closed under arbitrary unions of teams: If a formula of FO($\subseteq$) is true for each team in a finite or infinite collection $\{X_i : i \in I\}$, then the formula also holds for its union $\bigcup_{i\in I} X_i$. First-order logic (without dependency atoms) is both downwards closed and closed under union of teams, and thus has the *flatness property*: $\mathfrak{A} \models_X \psi$ if, and only if, $\mathfrak{A} \models_{\{s\}} \psi$ for all $s \in X$. Finally, independence logic is neither closed under unions of teams, nor downwards closed. For the counting constructs that we propose, we shall investigate whether or not they preserve such closure properties.

We will now introduce two methods for counting in team semantics that proved most fruitful during our work. After giving a formal definition, we will ensure that they fulfill the locality property and state their closure properties. Then, we will examine their expressive power in relation to other logics with team semantics, and fragments of Σ^1_1.

3 Forking

The *forking degree*, for two variable tuples $\overline{v}$ and $\overline{w}$, is defined as the number of different values assigned to $\overline{w}$ by all assignments with coinciding values for $\overline{v}$. A forking degree of one is equivalent to functional dependence so we can consider forking as a generalization of dependence. Note that this definition results in a separate forking degree for each group of assignments with the same value of $\overline{v}$. We introduce a counting mechanism based on *forking atoms* which, just as counting quantifiers, relate the counting result to a numeric variable.

▶ **Definition 2.** Forking atoms have the form $\overline{v} \lhd^{\leq \mu} \overline{w}$ and $\overline{v} \lhd^{\geq \mu} \overline{w}$, with the following semantics:

$$\mathfrak{A} \models_X \overline{v} \lhd^{\leq \mu} \overline{w} \iff \text{for all } s \in X, |\{s'(\overline{w}) : s' \in X,\ s'(\overline{v}) = s(\overline{v})\}| \leq s(\mu).$$

The definition for $\overline{v} \lhd^{\geq \mu} \overline{w}$ is analogous. When we extend a logic with both kinds of forking atoms, we can use the atom $\overline{v} \lhd^{=\mu} \overline{w}$ instead. Indeed, $\overline{v} \lhd^{=\mu} \overline{w} \equiv \overline{v} \lhd^{\leq \mu} \overline{w} \wedge \overline{v} \lhd^{\geq \mu} \overline{w}$ and, conversely, together with the ordering relation over numerals, we can simulate the two variants with $\overline{v} \lhd^{=\mu} \overline{w}$. We denote the extensions of FO by forking atoms of these forms

by FO(fork$^{\leq}$), FO(fork$^{\geq}$), and FO(fork$^{=}$). Since forking atoms only depend on values of explicitly mentioned variables, they trivally have the locality property, and this is preserved by all first-order operations.

▶ **Proposition 3.** *All extensions of* FO *by forking atoms satisfy the locality property.*

Whenever functional dependence is accessible in a logic, it is possible to define a bijection between a tuple $\overline{w}$ and a fresh numeric variable, so that we can reduce the forking atoms to a simpler form, with a single variable w on the right side rather than a tuple. We write FO(fork$_1^{\leq}$) for this restriction of FO(fork$^{\leq}$), and similarly for $\geq$ and $=$. Particularly, since a forking degree of one is equivalent to functional dependence, we can easily express dependence with the forking atom $\overline{v}\lhd^{\leq\mu}w$ and hence FO(fork$^{\leq}$) $\equiv$ FO(fork$_1^{\leq}$).

▶ **Example 4.** Regularity of graphs is definable in FO(fork$_1^{=}$), and even in FO(fork$_1^{\geq}$), by $\exists\mu\exists\nu(\mu+\nu=e\wedge\forall x\forall y((Exy\wedge x\lhd^{\geq\mu}y)\vee(\neg Exy\wedge x\lhd^{\geq\nu}y)))$. Recall the e is a numeric constant for the cardinality of the point sort.

Analogously to restricting dependence to constancy logic, there also is the special case of constant forking (abbreviated cfork), where the tuple $\overline{v}$ is of arity zero. The *constant forking atoms* $\lhd^{\leq\mu}\overline{w}$ and $\lhd^{\geq\mu}\overline{w}$ compare the number of different values of $\overline{w}$ with the value of μ.

▶ **Example 5** (Majority via constant forking)**.** One of the three possibilities, described in the introduction, for defining the team semantics of the majority quantifier was that $\mathfrak{A}\models_X My\varphi(\overline{v},y)$ if $\mathfrak{A}\models_{X[y\mapsto F]}\varphi$ for some function $F:X\to\mathcal{P}(A)$ with $|\bigcup_{s\in X}F(s)|\geq|A|/2$. This can equivalently be defined in FO(cfork$^{\geq}$) by the formula

$$\overline{v}=\overline{v}\vee(\exists\mu(\mu+\mu\geq e\wedge\exists y(\varphi(\overline{v},y)\wedge\lhd^{\geq\mu}y)).$$

To explain this, notice that the function $F:X\to\mathcal{P}(A)$ defines a split $X=Y\cup Z$ where $Y=\{s\in X:F(s)=\emptyset\}$ and $Z=\{s\in X:F(s)\neq\emptyset\}$, so that $X[y\mapsto F]=Z[y\mapsto F]$. Hence $\mathfrak{A}\models_{X[y\mapsto F]}\varphi$ means that $\mathfrak{A}\models_{Z[y\mapsto F]}\varphi$, and hence $\mathfrak{A}\models_Z\exists y\varphi$. Further $|\bigcup_{s\in X}F(s)|=|\bigcup_{s\in Z}F(s)|$. The displayed formula is a disjunction, and thus also imposes a split $X=Y\cup Z$, without any further restriction on Y (hence the disjunct $\overline{v}=\overline{v}$), and with the appropriate condition on Z. Notice the forking atom $\lhd^{\geq\mu}y$ implies in particular that Z is not empty.

One can also easily express in FO(fork$_1^{=}$) that two finite equivalence relations are isomorphic by saying that they have the same number of equivalence classes of any given size. Thus forking adds quite some expressiveness to first-order logic. It is easy to see, on the other side, that forking does not take us out of existential second-order logic (and NP). To examine the expressive power of forking more precisely it is useful to study the closure properties of forking logics.

▶ **Proposition 6.** *Forking atoms $\overline{v}\lhd^{\leq\mu}\overline{w}$ are downwards closed, but not under unions of teams, whereas atoms $\overline{v}\lhd^{\geq\mu}\overline{w}$ are closed under unions of teams but not downwards. Hence atoms $\overline{v}\lhd^{=\mu}\overline{w}$ are neither downwards closed nor closed under unions.*

As a direct consequence of the closure properties we obtain insights into the relationship between logics with forking and logics based on dependence atoms. Indeed, since dependence logic FO(dep) corresponds exactly to the downwards closed fragment of existential second-order logic [12] it must also contain FO(fork$^{\leq}$). By a more direct argument, we can, for two-sorted structures, directly translate a forking atom $\overline{v}\lhd^{\leq\mu}\overline{w}$ into $\exists\lambda(\lambda<\mu\wedge{=}(\overline{v}\lambda,\overline{w}))$. The idea of this formula is to extend the variables $\overline{v}$ with at most μ additional degrees of freedom and then demand a functional dependency between the extended tuple $\overline{v}\lambda$ and $\overline{w}$.

▶ **Theorem 7.** FO(dep) *is equivalent to* FO(fork$^{\leq}$)*, and incomparable to* FO(fork$^{\geq}$)*. Finally,* FO(fork$^{=}$) *is strictly stronger than* FO(dep)*.*

FO(fork$^{\geq}$) is incomparable with FO(dep), having the same closure properties as inclusion logic. However, FO(fork$^{\geq}$) is not equivalent with inclusion logic. For instance, by a simple Ehrenfeucht-Fraïssé argument one can show that regularity of graphs is not expressible in LFP, and hence neither in inclusion logic, but we have seen above that it is definable in FO(fork$^{\geq}$). We conjecture that FO(fork$^{\geq}$) and FO($\subseteq$) are incomparable.

It remains to determine the expressive power of the strongest forking logic FO(fork$^{=}$). It turns out that, even when restricted to constant forking, it has the full power of independence logic, and thus of existential second-order logic and NP.

▶ **Theorem 8.** *On two-sorted structures,* FO(cfork$^{=}$) $\equiv$ FO(fork$^{=}$) $\equiv$ FO(indep)*.*

Proof. Since FO(indep) has the full power of Σ^1_1, we just have to prove that any independence atom is equivalent to a formula in FO(cfork$^{=}$). We claim that

$$\overline{v}\perp\overline{w} \equiv \exists\mu\exists\nu(\triangleleft^{=\mu}\overline{v} \wedge \triangleleft^{=\nu}\overline{w}) \wedge \triangleleft^{\mu\nu}\overline{vw}.$$

Indeed, it is not difficult to verify that the two formulae are just two different ways to express that $X[\overline{vw}] = X[\overline{v}] \times X[\overline{w}]$. There is an alternative way to express independence by non-constant forking, but with just one counting variable. Indeed,

$$\overline{v}\perp\overline{w} \equiv \exists\mu(\triangleleft^{=\mu}\overline{w} \wedge \overline{v}\triangleleft^{=\mu}\overline{w}).$$

For any team X, let $Y := X[\mu \mapsto m]$ for $m = |X[\overline{w}]|$. Assume that $\models_X \overline{v}\perp\overline{w}$. Clearly, $\models_Y \triangleleft^{=\mu}\overline{w}$. To prove that also $\overline{v}\triangleleft^{=\mu}\overline{w}$ holds in Y we have to show that any value $\overline{a} \in Y[\overline{v}] = X[\overline{v}]$ forks to *all* values $\overline{b} \in Y[\overline{w}] = X[\overline{w}]$. Fix $s, s' \in X$ with $s(\overline{v}) = \overline{a}$ and $s'(\overline{w}) = \overline{b}$. By $\overline{v}\perp\overline{w}$ there exists an assignment $s'' \in X$ with $s''(\overline{v}) = \overline{a}$ and $s''(\overline{w}) = \overline{b}$. Thus $t'' = s''[\mu \mapsto m] \in Y$ witnesses the forking of $\overline{a}$ to $\overline{b}$.

Conversely, assume that $\models_X \exists\mu(\triangleleft^{=\mu}\overline{w} \wedge \overline{v}\triangleleft^{=\mu}\overline{w})$, which implies that $\models_Y \overline{v}\triangleleft^{=\mu}\overline{w}$. Choose two assignment $s, s' \in X$. Since $s(\overline{v})$ forks in Y to all values in $Y[\overline{w}]$, it forks in particular to $s'(\overline{w})$, so there exists an assignment $t'' \in Y$ with $t''(\overline{v}) = s(\overline{v})$ and $t''(\overline{w}) = s'(\overline{w})$. The restriction of t'' to the domain of X thus witnesses the truth of $\overline{v}\perp\overline{w}$. ◀

Forking atoms thus provide a lot of power. Compared to the familiar logics with team semantics such as dependence and independence logic, which are able to express NP-complete problems but usually in a somewhat roundabout way that is rather hard to find and to read, forking atoms often lead to more direct and natural definitions. Further, the two-sorted framework makes it much easier to deal with problems that include numeric parameters, such as bounds on the size of solutions or structures with weights. To illustrate this, we consider some familiar NP-complete problems.

▶ **Example 9** (Dominating Set, Vertex Cover, and Clique)**.** The dominating set problem can be expressed in FO(cfork$^{\leq}$). Indeed, a graph $G = (V, E)$ admits a dominating set of size k if, and only if, $(G, k) \models \forall y\exists x(Exy \wedge \triangleleft^{\leq k}x)$. A similar idea works for Vertex Cover and Clique. A graph G has a vertex cover of size $\leq k$ if $(G, k) \models \forall x\forall y(\neg Exy \vee \exists z((z = x \vee z = y) \wedge \triangleleft^{\leq k}z)$, and it has a clique of size $\geq k$ if $(G, k) \models \forall x\forall y(Exy \vee \exists z((z = x \vee z = y) \wedge \triangleleft^{\leq e-k}z)$. These examples also separate FO(cfork$^{\leq}$) from FO(const), since for sentences the latter collapses to FO.

▶ **Example 10** (TSP). A more ambitious challenge for the use of forking atoms is a presentation of the TSP. An instance D of the TSP, with distances $d_{ij} \in \mathbb{N}$, for $i, j \in \{0, \dots, n-1\}$ with $i \neq j$, can be represented as a team $X(D)$ consisting of the $n(n-1)$ assignments $s_{ij} : (x, y, \lambda) \mapsto (i, j, d_{ij})$ into A^*, for $A = \{a_0, \dots, a_{n-1}\}$. We construct a formula $\psi(x, y, \lambda, k) \in \mathrm{FO}(\mathrm{fork}^{\leq})$ such that, for all such A, $X(D)$ and all $k \in \omega$,

$$(A, k) \models_{X(D)} \psi \iff D \text{ admits a TSP-tour of length} \leq k.$$

We first notice that for every formula φ such that $(A, k) \models_{X(D)} x \triangleleft^{\leq e-2} y \vee (y \triangleleft^{\leq 1} x \wedge \varphi)$ it follows that φ must hold in some subteam $Z \subseteq X(D)$ that covers A by a disjoint union of cycles. More precisely, this means that the directed graph (A, E_Z) with $E_Z = \{(a_i, a_j) : d_{ij} \in Z\}$ consists of disjoint cycles, and every a_i occurs in precisely one of these cycles. Indeed the formula requires a split $X(D) = Y \cup Z$, such that Y contains, for every i, at most $n-2$ assignment s_{ij}. Thus, for each i there remains at least one assignment s_{ij} that must be in Z, but on the other side, by the forking atom $y \triangleleft^{\leq 1} x$ at most one s_{ij} can be in Z for each j. The only way to satisfy these contraints is that E_Z defines a bijection (and thus a covering by disjoint cycles). It remains to construct φ so that it enforces, for such a subteam Z, that it in fact consists of just one cycle, and that the length of this cycle does not exceed k. This is achieved by

$$\begin{aligned}\varphi := \exists c(\triangleleft^{\leq 1} c \wedge \exists \mu \exists \nu \forall x' \exists \mu' \exists \nu' (x \triangleleft^{\leq 1} \mu \wedge x' \triangleleft^{\leq 1} \mu' \wedge x \triangleleft^{\leq 1} \nu \wedge x' \triangleleft^{\leq 1} \nu' \wedge \\ (x = x' \to \mu = \mu' \wedge \nu = \nu') \wedge (x = c \to \mu = 0 \wedge \nu = 1) \wedge \\ (x' = y \neq c \to \mu' = \mu + \lambda \wedge \nu' = \nu + 1) \wedge (x' = y = c \to \mu + \lambda \leq k \wedge \nu = e)))\end{aligned}$$

Recall that in the team Z we have for every i precisely one assigment $s_{ij} \in Z$. The quantifiers and the first two lines of the formula thus imply that there is a node c and functions $a_i \mapsto \mu_i$ and $a_i \mapsto \nu_i$ assigning to each node two numbers. Viewing c as the beginning of the tour, and a_i as a node on the same cycle as c, it follows by induction on the path from c to a_i that φ imposes that the value of μ_i is the length of the path from c to a_i, and that the value of ν_i is the number of nodes on that path. Finally for the closing of the cycle at c, the formula says that the cycle has length $\leq k$ and contains all nodes. Thus, the subteam Z must indeed be a TSP tour of length at most k.

Since constant forking is a restriction of general forking, it inherits all locality and closure properties. Furthermore, also the relationship between functional dependence and forking can be readily translated to constancy logic and constant forking. However, in contrast to the equivalence between FO(dep) and FO(fork$^{\leq}$) it is possible to separate the constant variant FO(cfork$^{\leq}$) from FO(const). See Figure 1, at the end of this paper, for an illustration and summary of such results.

4 Counting Quantifiers

Counting quantifiers of form $\exists^{\geq \mu} x$ provide a well-known and powerful way to add counting to logics with Tarski semantics, such as first-order logic or fixed-point logic. We adapt them to team semantics, and investigate the properties and expressive power of the resulting extensions. In fact, we propose a rather general variant of such quantifiers which admits not only the counting of single elements (over the point sort) but counting of arbitrary tuples, even of mixed type.

▶ **Definition 11.** *Counting quantifiers* for teams permit to build, for every formula φ, every numeric variable μ, every closed numeric term t, and every tuple $\overline{v}$ of variables of mixed type,

the new formula $\exists^{\geq\mu}\overline{v}_{<t}\varphi$. For any structure $\mathfrak{A}^*$ and every team X whose domain includes μ and all variables in $\mathrm{free}(\varphi) \setminus \{\overline{v}\}$, we have that $\mathfrak{A}^* \models_X \exists^{\geq\mu}\overline{v}_{<t}\varphi$ if, and only if, there is a function F that maps every $s \in X$ to a set $F(s)$ of at least $s(\mu)$ many, appropriately typed, tuples over $A \cup \omega$, whose numeric components are bounded by t, such that $\mathfrak{A}^* \models_{X[\overline{v}\mapsto F]} \varphi$.

▶ **Example 12** (Majority via counting quantifiers)**.** If we define the team semantics of the majority quantifier so that $\mathfrak{A} \models_X My\varphi(\overline{v}, y)$ if $\mathfrak{A} \models_{X[y\mapsto F]} \varphi$ for a function $F : X \mapsto \mathcal{P}(A)$ with $|F(s)| \geq |A|/2$ for all $s \in X$, then this is equivalent to $\exists\mu(\mu + \mu \geq e \wedge \exists^{\geq\mu}y\varphi(\overline{v}, y))$. If we require instead that $F(s) = B$ for some fixed set B with $|B| \geq |A|/2$, then we need a combination of counting quantifiers and forking atoms, namely $\exists\mu(\mu+\mu \geq e \wedge \exists^{\geq\mu}y(\varphi(\overline{v}, y) \wedge \lhd^{\leq\mu}y))$.

▶ **Proposition 13.** *Counting quantifiers preserve the locality property.*

Notice instead that counting quantifiers of form $\exists^{\leq\mu}y$ or $\exists^{=\mu}y$ that are in common use in logics with Tarski semantics, are unsafe for team semantics since they may violate the locality principle. Indeed, even the quantifiers $\exists^{\leq 1}y$ or $\exists^{=1}y$ can be used to simulate the strict semantics for common existential quantifiers, and we have already seen that this is in conflict with the locality principle. Observe that, in contrast to forking, counting quantifiers do not express any dependency between assignments.

▶ **Proposition 14.** *Counting quantifiers preserve downwards closure and closure under unions. As a consequence,* FO *with counting quantifiers, but without any dependence atoms, is flat.*

Nevertheless, FO with counting is more expressive than without counting. It can express for instance even cardinality or regularity of graphs. We already mentioned in the introduction that counting is definable in logics whose expressive power comes sufficiently close to Σ^1_1. In particular, this is the case for dependence logic (but not, for instance, for inclusion logic). In fact dependence statements can be seen as a particular form of weak counting, and we show that these suffice, over two-sorted structures, to express counting quantifiers in a rather simple way.

▶ **Proposition 15.** *Counting quantifiers are expressible by means of dependence atoms.*

Proof. A formula $\psi := \exists^{\geq\mu}y\varphi$ is equivalent to $\forall\lambda(\lambda \geq \mu \vee \exists y(=(\mathrm{free}(\psi)y, \lambda) \wedge \varphi))$. This construction readily extends to more general counting quantifiers. ◀

5 Inclusion Logic with Counting Quantifiers

We now study the logic $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ that extends inclusion logic with counting quantifiers, and show that it is equivalent, in the sense described in the introduction, with fixed-point logic with counting (FPC). Recall that for all two-sorted structures $\mathfrak{A}^*$ we always require the point sort $\mathfrak{A}$ to be a finite structure.

5.1 Fixed-point logic with counting with only greatest fixed-points

As already mentioned, FPC is usually defined as the extension of first-order logic over two-sorted structures by counting terms and inflationary fixed-points. However, this definition is not really adequate for proving an equivalence with a logic with team semantics, for two reasons: Counting terms may violate the locality principle, and inflationary fixed-points

have no direct translation into logics with team semantics (or into Σ^1_1, for that matter)[3]. We therefore work with a different definition of FPC, that is based on greatest fixed-point operators, used only positively, and on counting quantifiers of form $\exists^{\geq\mu}\overline{v}$. Of course, we have to convince ourselves that this syntactically restricted variant is semantically equivalent, i.e. that we do not lose expressive power. While this is not trivial, it can be proved by combining some well-understood techniques.

First of all, counting terms $\#_x\varphi(x)$ can readily be replaced by formulae with counting quantifiers of the form $\exists^{=\mu}x\varphi$, and these can be rewritten by $\exists^{\geq\mu}x\varphi(x) \wedge \exists\nu(\mu + \nu = e \wedge \exists^{\geq\nu}x\neg\varphi(x))$. However, this introduces negation, which poses a problem with monotonicity, and this is one of the reasons why one normally prefers inflationary fixed-points rather than least and greatest ones.

In the absence of counting it is known that, by means of the Stage Comparison Theorem, one can eliminate inflationary fixed points and prove that the logics LFP and IFP are semantically equivalent [8, 13]. On finite structures, one can even go an important step further and prove (again based on the Stage Comparison Theorem) that the negation of a greatest fixed point is equivalent to a formula using greatest fixed points only positively [11].

It is straightforward (but a bit lengthy) to verify that the proof of the Stage Comparison Theorem, and also its applications, go through for the case of two-sorted structures and in the presence of counting quantifiers. Thus we can indeed, without loss of generality, assume that all formulae in FPC are written in this restricted form. In particular, this has the advantage that we have a relatively simple description of model-checking games for FPC, as so-called threshold safety games.

5.2 Threshold games

A threshold game is a two-player game on a finite (or at least finitely branching) directed graph $G = (V, E)$ equipped with a threshold function $\theta : V \to \omega$. Let $vE = \{w : (v, w) \in E\}$ and let $\delta(v) := |vE|$ denote the out-degree of v. We assume that $\theta(v) \leq \delta(v) + 1$ for all v.

At any given node v in a play, Player 0 selects a set $X \subseteq vE$ of $\theta(v)$ successors of v, then Player 1 choses a node $w \in X$ and the play proceeds from w. When a player cannot move, she loses. This means that Player 0 wins at all nodes in $T_0 := \{v \in V : \theta(v) = 0\}$, and Player 1 wins at nodes in $T_1 := \{v \in V : \delta(v) < \theta(v)\}$.

Classical (finitely branching) graph games, where the set of nodes is partitioned into two sets, $V = V_0 \cup V_1$, such that Player 0 moves from nodes in V_0 and Player 1 from those in V_1, can be viewed as the special case of threshold games where, for all nodes v, either $\theta(v) = 1$ or $\theta(v) = \delta(v)$. In principle, we can combine threshold games with any winning condition for infinite plays, but in this paper we just consider *threshold safety games*, where Player 0 just has to avoid the positions in T_1 where she loses immediately. In particular, Player 0 wins all infinite plays.

The winning region of a player is the set of positions from which she has a winning strategy. The well-known linear-time algorithm for computing winning regions in classical reachability and safety games can be adapted to threshold games.

▶ **Proposition 16.** *The winning regions of a threshold safety game $\mathcal{G} = (V, E, \theta : V \to \omega)$ on a finite game graph can be computed in time $O(|V| + |E|)$.*

[3] On finite structures we have an indirect translation into Σ^1_1 since fixed-point logics are in polynomial time and Σ^1_1 captures NP, but on infinite structures, LFP and IFP are on the Δ^1_2-level.

Even more relevant for us is the relationship to logics with counting. We first note that the winning region W_0 for Player 0 in threshold safety games $\mathcal{G} = (V, E, \theta : V \to \omega)$ is uniformly definable by a very simple formula in fixed-point logic with counting FPC, namely $\mathrm{win}(x) := [\mathbf{gfp}\, Wx\,.\, \exists^{\geq\theta(x)} y(Exy \wedge Wy)](x)$. Here and in the following, $\exists^{\geq\theta(x)} y\varphi(y)$ is just an abbreviation for $\exists\mu(\mu = \theta(x) \wedge \exists^{\geq\mu} y\varphi)$. Equivalently, the winning region is definable in inclusion logic with counting, by $\mathrm{win}'(x) := \exists^{\geq\theta(x)} y(Exy \wedge y \subseteq x)$. Then for any threshold game graph $\mathcal{G}$, the maximal team W such that $\mathcal{G} \models_W \mathrm{win}'(x)$ is precisely the winning region for Player 0.

For the translation between the two logics, we shall use a specific kind of trap condition for initial positions.

▶ **Definition 17.** Fix a set $I \subseteq V$ of initial positions in a threshold safety game $\mathcal{G}$. An *I-trap* in $\mathcal{G}$ is a set $Z \subseteq I$ such that Player 0 has a winning strategy from Z that, moreover, avoids $I \setminus Z$.

Such a winning condition can simply be described by a subset $W \subseteq V$ such that $W \cap I = Z$, and $|vE \cap W| \geq \theta(v)$ for all $v \in W$. Indeed, at any position $v \in W$, Player 0 can then select any set $X \subseteq vE \cap W$ with $|X| = \theta(v)$ and wins since $W \cap (I \setminus Z) = \emptyset$ and $W \cap T_1 = \emptyset$. A straightforward modification of the formulae for winning regions shows that in both logics, FPC and $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$, we can define also I-traps.

But the connection between threshold games and logics with counting goes much deeper. Threshold safety games arise as the model-checking games for both inclusion logic with counting and FPC, and moreover, the model-checking games are uniformly interpretable in the structure in which the formulae are evaluated.

5.3 Model-checking games and game interpretations

It is known, and explained in detail in [6], how to construct classical safety games as evaluation games for inclusion logic and for the posGFP fragment of fixed-point logic. We extend these constructions to obtain threshold safety games as model-checking games, on finite structures, for inclusion logic with counting and for FPC.

We first sketch the construction of a threshold safety game $\mathcal{T}(\mathfrak{A}^*, \psi)$ for a two-sorted structure $\mathfrak{A}^*$ and a formula $\psi(\overline{x}, \overline{\mu})$ of fixed-point logic with counting (which uses only greatest fixed points). Let $G(\psi) = (\mathrm{Sf}(\psi), E_\psi)$ be the syntax graph of ψ. Positions of the game are pairs (φ, s) where $\varphi \in \mathrm{Sf}(\psi)$ is a subformula of ψ and $s : \mathrm{free}(\varphi) \to (A \cup \omega)$ is an (appropriately typed) assignment on the free variables of φ. The immediate successors of a position (φ, s), are the pairs (φ', s') where $(\varphi, \varphi') \in E_\psi$ and s and s' coincide on the common variables. A position $([\mathbf{gfp}\, Z\overline{x}\overline{\mu}_{<t}\,.\, \eta(Z, \overline{x}, \overline{\mu})](\overline{x}, \overline{\mu}), s)$ has the unique successor (η, s). For any fixed-point atom $Z\overline{y}\overline{\nu}$ in the scope of η, the unique successor of a position $(Z\overline{y}\overline{\nu}, s)$ is (η, s') with $s'(\overline{x}, \overline{\mu}) = s(\overline{y}, \overline{\nu})$. Thresholds $\theta(\varphi, s)$ are assigned as follows:

1. In the case that φ is a first-order literal, we set $\theta(\varphi, s) := 0$ if $\mathfrak{A} \models_s \varphi$ and $\theta(\varphi, s) := 1$ if $\mathfrak{A} \not\models_s \varphi$.
2. We set $\theta(\varphi, s) := 1$ in all cases where φ is either a fixed-point atom $Z\overline{x}\overline{\mu}$, a fixed-point formula $[\mathbf{gfp}\, Z\overline{x}\overline{\mu}_{<t}\,.\, \eta(Z, \overline{x}, \overline{\mu})](\overline{x}, \overline{\mu})$, a disjunction $\varphi_1 \vee \varphi_2$, or an existentially quantified formula $(\exists\mu < t)\varphi'$.
3. We set $\theta(\varphi, s) := 2$ if φ is a conjunction $\varphi_1 \wedge \varphi_2$.
4. For formulae $\varphi := (\forall\mu < t)\varphi'$, we set $\theta(\varphi, s) = t^{\mathfrak{A}^*}$, i.e. the value of the numeric term t in $\mathfrak{A}^*$
5. For formulae with counting quantifiers $\varphi := \exists^{\geq\mu}\overline{v}\varphi'$, we put $\theta(\varphi, s) = s(\mu)$.

▶ **Theorem 18.** *For every structure $\mathfrak{A}^*$, every formula $\psi(\overline{x},\overline{\mu}) \in \mathrm{FPC}$, and every appropriately typed assignment $s : \mathrm{free}(\varphi) \to (A \cup \omega)$, we have that $\mathfrak{A}^* \models_s \psi(\overline{x},\overline{\mu})$ if, and only if, Player 0 has a winning strategy for the threshold safety game $\mathcal{T}(\mathfrak{A}^*,\psi)$ from position (ψ, s).*

For the relationship with inclusion logic we shall need a more refined result, concerning sentences in FPC of vocabulary $\tau \cup \{X\}$ of the form $\psi := \forall\overline{x}\forall\overline{\mu}(X\overline{x\mu} \to \varphi(\overline{x},\overline{\mu}))$, such that X occurs only positively in φ. In that case, the model checking game $\mathcal{T}((\mathfrak{A}^*, X), \psi)$ is a threshold safety game with unique initial position $(\psi, \emptyset)$. We modify these games by eliminating the explicit reference to the relation X and associate the model checking problem of whether $(\mathfrak{A}, X) \models \psi$ with a trap condition for a modified game $\mathcal{T}^{\#}(\mathfrak{A}^*, \varphi)$. To do this, we identify every position of form $(X\overline{y\nu}, t)$ with the position $(\varphi(\overline{x},\overline{\mu}), s)$ such that $s(\overline{x},\overline{\mu}) = t(\overline{y},\overline{\nu})$; this means that every edge in the game graph to a position $(X\overline{y\nu}, t)$ is replaced by an edge to $(\varphi(\overline{x},\overline{\mu}), s)$, and the node $(X\overline{y\nu}, t)$ is deleted. The set I of initial positions now consists of all pairs of form $(\varphi(\overline{x},\overline{\mu}), s)$. Given any interpretation for the relation X, let $X^* \subseteq I$ be the set of positions (φ, s) where $s(\overline{x},\overline{\mu}) \in X$.

▶ **Proposition 19.** *$(\mathfrak{A}^*, X) \models \forall\overline{x}\forall\overline{\mu}(X\overline{x\mu} \to \varphi(\overline{x},\overline{\mu}))$ if, and only if, X^* is an I-trap in $\mathcal{T}^{\#}(\mathfrak{A}^*, \varphi)$.*

The construction for inclusion logic with counting is similar. It extends the construction given in [6] of safety games for $\mathrm{FO}(\subseteq)$. With every formula $\psi(\overline{x},\overline{\mu})$ in $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ and every structure $\mathfrak{A}^*$, we construct a threshold safety game $\mathcal{T}(\mathfrak{A}^*,\psi)$, played on a forest of game trees, where again, positions are pairs (φ, s) where φ is an occurrence of a subformula in ψ and s an assignment with domain $\mathrm{free}(\varphi)$. The set I of initial positions is the set of roots of the game trees in $\mathcal{T}(\mathfrak{A}^*,\psi)$; it contains all pairs (ψ, s). Hence, a team X for ψ defines the subset $I(X) := \{(\psi, s) : s \in X\}$ of I. Instead of the regeneration of fixed-points we here have a regeneration mechanism for inclusion atoms. The threshold game graph is set up so that, informally, at a position $((\overline{x\mu} \subseteq \overline{y\nu}), s)$ associated with an inclusion atom, Player 0 selects an assignment t such that $t(\overline{y},\overline{\nu}) = s(\overline{x},\overline{\mu})$, moves to $((\overline{x\mu} \subseteq \overline{y\nu}), t)$ and then Player 1 takes the play to an ancestor of that node in the corresponding game tree. (A more formal construction requires that we duplicate all nodes in the game, so that the moves going upwards in the game tree actually take place in a separate copy of the tree. For details of this, in the context of classical safety games for inclusion logic, see [6]). To win, Player 0 has to make sure that the play remains inside those trees with a root in $I(X)$, which means that these roots form an I-trap.

▶ **Theorem 20.** *For every formula $\psi(\overline{x},\overline{\mu}) \in \mathrm{FO}(\subseteq, \exists^{\geq\mu})$, every structure $\mathfrak{A}^*$ and every team X, we have that $\mathfrak{A}^* \models_X \psi(\overline{x},\overline{\mu})$ if, and only if, $I(X)$ is an I-trap in $\mathcal{T}(\mathfrak{A}^*,\psi)$.*

Proof. Suppose that $\mathfrak{A}^* \models_X \psi(\overline{x},\overline{\mu})$. Then, according to the rules defining the semantics of ψ, we can assign to every occurrence of a subformula φ in ψ a team $Y(\varphi)$ such that $Y(\psi) = X$ and $\mathfrak{A}^* \models_{Y(\varphi)} \varphi$ for all φ. In particular, for $\varphi = \varphi_1 \vee \varphi_2$ we have that $Y(\varphi) = Y(\varphi_1) \cup Y(\varphi_2)$, for $\varphi = \exists^{\geq\mu}\overline{v}\eta$ we have that $Y(\varphi) = Y(\eta)[\overline{v} \mapsto F]$ for a function $F : Y(\eta) \to \mathcal{P}((A^*)^k)$ with $|F(s)| \geq s(\mu)$, and so on for the other types of formulae. We define a strategy W for Player 0 in $\mathcal{T}(\mathfrak{A}^*,\psi)$ by

$$W := \{(\varphi, s) : \varphi \in \mathrm{Sf}(\psi), s \in Y(\varphi)\}.$$

It is straightforward to verify that for any position $(\varphi, s) \in W$, there are at least $\theta(\varphi, s)$ many successors of (φ, s) inside W, and if $(\varphi, s) \notin W$, then none of the nodes in the subtree rooted at (φ, s) belongs to W. In particular, for any inclusion atom $\alpha := (\overline{x\mu} \subseteq \overline{y\nu})$ and any

position of form $(\alpha, s) \in W$ we infer that, since α is true in $Y(\alpha)$ there exist an assignment $t \in Y(\alpha)$ with $t(\overline{y}, \overline{\nu}) = s(\overline{x}, \overline{\mu})$, so Player 0 can indeed take the game to an occurrence of (α, t) in W. Further, since W only contains nodes in trees with a root (ψ, s) such that $s \in X$, Player 1 can force the game, by going upwards the game trees, only to initial positions in $I(X)$. Thus $I(X)$ is indeed an I-trap in $\mathcal{T}(\mathfrak{A}^*, \psi)$.

Conversely, suppose that W describes a strategy for Player 0 showing that $I(X)$ is an I-trap in $\mathcal{T}(\mathfrak{A}^*, \psi)$. For every node (φ, s) in $\mathcal{T}(\mathfrak{A}^*, \psi)$, let

$$\mathrm{Team}(W, \varphi) := \{s : (\varphi, s) \in W\}.$$

In particular, $\mathrm{Team}(W, \psi) = X$. By induction on the syntax of ψ, one easily verifies that for every $\varphi \in S(\psi)$,

$$\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi)} \varphi.$$

We just discuss the most interesting cases.

- If φ is a first-order literal, then all nodes (φ, s) are terminal nodes, so W can contain (φ, s) only if $\theta(\varphi, s) = 0$ which is the case if, and only if, $\mathfrak{A}^* \models_s \varphi$. Thus $\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi)} \varphi$.
- Let φ be an inclusion atom $(\overline{x\mu} \subseteq \overline{y\nu})$. For all s such that $(\varphi, s) \in W$, there exists an assignment t with $t(\overline{y}, \overline{\nu}) = s(\overline{x}, \overline{\mu})$ such that also $(\varphi, t) \in W$. But this means that $\mathrm{Team}(W, \varphi)$ satisfies $(\overline{x\mu} \subseteq \overline{y\nu})$.
- If $\varphi = \varphi_1 \vee \varphi_2$, then for every node $(\varphi, s) \in W$, at least one of its two successors must also belong to W. Let s_1 and s_2 be the restrictions of s to the free variables of φ_1 and φ_2, respectively, and let Y_i be the team of all assignments s with domain $\mathrm{free}(\varphi)$ such that $s_i \in \mathrm{Team}(W, \varphi_i)$. It follows that $\mathrm{Team}(W, \varphi) = Y_1 \cup Y_2$. By induction hypothesis, we have that $\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi_i)} \varphi_i$ for $i = 1, 2$ and, by the locality principle, also $\mathfrak{A}^* \models_{Y_i} \varphi_i$. It follows that $\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi)} \varphi$.
- For $\varphi = \exists^{\geq\mu}\overline{v}\eta$ we have that $\theta(\varphi, s) = s(\mu)$. Thus, if $(\varphi, s) \in W$ then there exist at least $s(\mu)$ many assignments $t = s[\overline{v} \mapsto \overline{c}]$ such that $(\eta, t) \in W$, and hence $t \in \mathrm{Team}(W, \eta)$. By induction hypothesis, $\mathfrak{A}^* \models_{\mathrm{Team}(W,\eta)} \eta$ which implies that there is a function $F : \mathrm{Team}(W, \varphi) \to \mathcal{P}(A^*)^k$ with $|F(s)| \geq s(\mu)$ for all $s \in \mathrm{Team}(W, \varphi)$ and $\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi)[\overline{v}\mapsto F]} \eta$. But this means that $\mathfrak{A}^* \models_{\mathrm{Team}(W,\varphi)} \varphi$.

The remaining cases are routine. Since $\mathrm{Team}(W, \psi) = X$ this implies that $\mathfrak{A}^* \models_X \psi$. ◀

Observe that, for any fixed formula $\psi(\overline{x}, \overline{\mu})$ in any of these logics, the construction of the threshold safety games is done in a very uniform way. This intuition is made precise by the notion of a *game interpretation*. In our case, such an interpretation is a quadruple $J = (\delta, e, \mathrm{in}, \vartheta)$ consisting of first-order formulae $\delta(\lambda, \overline{v})$, $e(\lambda, \overline{v}; \lambda', \overline{v}')$, and $\mathrm{in}(\lambda, \overline{v})$ for the nodes, edges, and initial positions of the game graph, and a formula $\vartheta(\lambda, \overline{v}, \mu)$ for the thresholds. Given a structure $\mathfrak{A}^*$, we obtain a graph whose set of nodes is $\delta^{\mathfrak{A}^*} := \{(i, \overline{c}) : \mathfrak{A}^* \models \delta(i, \overline{c})\}$ and whose sets of edges $e^{\mathfrak{A}^*}$ and initial positions $\mathrm{in}^{\mathfrak{A}^*}$ are defined in an analogous way.

We say that J interprets the threshold game $\mathcal{T} = (V, E, I, \theta : V \to \omega)$ in $\mathfrak{A}^*$, if there is an isomorphism $h : (\delta^{\mathfrak{A}^*}, e^{\mathfrak{A}^*}, \mathrm{in}^{\mathfrak{A}^*}) \to (V, E, I)$, such that thresholds are defined by $\vartheta(\lambda, \overline{v}, \mu)$ as follows: For every node $v \in V$ and every number $t \in \omega$ we have that $\mathfrak{A}^* \models \vartheta(h^{-1}(v), t)$ if, and only if, $\theta(v) = t$.

▶ **Theorem 21.** *For every formula $\psi(\overline{x}, \overline{\mu})$ in $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ there is a first-order interpretation $J(\psi)$ that interprets, for any structure $\mathfrak{A}^*$, the game $\mathcal{T}(\mathfrak{A}^*, \psi)$ in $\mathfrak{A}^*$.*

Proof. For simplicity, we assume that the vocabulary of ψ and $\mathfrak{A}$ contains a constant c. We enumerate the subformulae of ψ as $\varphi_0, \dots, \varphi_\ell$, with $\varphi_0 = \psi$, and assume that $\overline{v}$ is the tuple of all variables occurring in ψ. For every formula φ_i we extend every assignment $s : \mathrm{free}(\varphi_i) \to A^*$ to an assignment s^* on all variables in $\overline{v}$ by setting $s^*(y) = c$ and $s^*(\nu) = 0$ for all variables y or ν that appear in $\overline{v}$ but not in $\mathrm{free}(\varphi_i)$. The interpretation $J(\psi)$ then represent a position (φ_i, s) of $\mathcal{T}(\mathfrak{A}^*, \psi)$ by the tuple $(i, s^*(\overline{v}))$ in A^*. With this representation, it is completely straightforward to construct quantifier-free formulae $\delta(\lambda, \overline{v})$, $e(\lambda, \overline{v}; \lambda', \overline{v}')$, $\mathrm{in}(\lambda, \overline{v})$, and $\vartheta(\lambda, \overline{v}, \mu)$ with the required properties. ◀

Analogous statements hold for the games for FPC.

5.4 Game-based translations between $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ and FPC

A first-order interpretation J may be seen as a function mapping a structure $\mathfrak{C}$ to the interpreted structure $J(\mathfrak{C})$, and by the coordinate map h of J, every element $b \in J(\mathfrak{C})$ is associated with a tuple $h^{-1}(b) \in \mathfrak{C}$. But in the other direction, J also gives a translation from formulae $\varphi(x_1, \dots, x_k)$ over $J(\mathfrak{C})$ to formulae $\varphi^J(\overline{x}_1, \dots, \overline{x}_k)$ over $\mathfrak{C}$ such that

$$J(\mathfrak{C}) \models \varphi(b_1, \dots, b_k) \iff \mathfrak{C} \models \varphi^J(h^{-1}(b_1), \dots, h^{-1}(b_k)).$$

This is called the Interpretation Lemma, which is of course a general and well-known fact that holds (in appropriate form) for all kinds of interpretations, not just the particular form of game interpretations that we use here.

We are now ready to prove the main theorem of this paper. We shall combine the interpretation argument for threshold safety games and the definability of I-traps, by means of the Interpretation Lemma, to provide effective translations between the two logics.

▶ **Theorem 22.** *There exist effective translations in both directions between formulae $\psi(\overline{x}, \overline{\mu})$ in $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$, and formulae $\varphi(X, \overline{x}, \overline{\mu})$ in* FPC *(with only positive occurrences of X), such that, for every structure $\mathfrak{A}^*$ and every X,*

$$\mathfrak{A}^* \models_X \psi(\overline{x}, \overline{\mu}) \iff (\mathfrak{A}^*, X) \models \forall\overline{x}\forall\overline{\mu}(X\overline{x}\overline{\mu} \to \varphi(X, \overline{x}, \overline{\mu})).$$

In particular, on every structure $\mathfrak{A}^*$, the maximal team satisfying ψ coincides with the greatest fixed-point of φ. For sentences, $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ and FPC *have the same expressive power.*

Proof. We first describe how to translate $\psi(\overline{x}, \overline{\mu}) \in \mathrm{FO}(\subseteq, \exists^{\geq\mu})$ into an appropriate formula $\varphi(X, \overline{x}, \overline{\mu})$ in FPC.

Take a formula in FPC that defines I-traps in threshold games, so that in particular

$$(\mathcal{T}(\mathfrak{A}^*, \psi), Z) \models \forall x(Zx \to \mathrm{itrap}(Z, x)) \iff Z \text{ is an I-trap in } \mathcal{T}(\mathfrak{A}^*, \psi).$$

The interpretation $J(\psi)$ defines a copy of $\mathcal{T}(\mathfrak{A}^*, \psi)$ inside $\mathfrak{A}^*$ and maps formulae on the game back to formulae on $\mathfrak{A}^*$. Thus, $J(\psi) : \mathrm{itrap}(Z, x) \mapsto \mathrm{itrap}^*(Y, \overline{y}, \overline{\nu})$. If $Y \subseteq A^k \times \omega^\ell$ is the set of tuples in $\mathfrak{A}^*$ associated with $Z \subseteq I$, then $(\mathfrak{A}^*, Y) \models \forall\overline{y\nu}(Y\overline{y\nu} \to \mathrm{itrap}^*(Y, \overline{y}, \overline{\nu}))$ if, and only if, Z is an I-trap in $\mathcal{T}(\mathfrak{A}^*, \psi)$.

The tuples describing positions (ψ, s) with $s \in X$ are first-order definable in $(\mathfrak{A}^*, X)$. We can thus massage $\mathrm{itrap}^*(Y, \overline{y}, \overline{\nu})$ into a formula $\varphi(X, \overline{x}, \overline{\mu}) \in \mathrm{FPC}$ such that

$$(\mathfrak{A}^*, X) \models \forall\overline{x\mu}(X\overline{x\mu} \to \varphi(X, \overline{x}, \overline{\mu})) \iff I(X) \text{ is an I-trap in } \mathcal{T}(\mathfrak{A}^*, \psi) \iff \mathfrak{A} \models_X \psi.$$

An analogous construction works for the translation of FPC into $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$, making use of the fact that I-traps are definable also in inclusion logic with counting. Let $\mathcal{T}^\#(\mathfrak{A}^*, \varphi)$

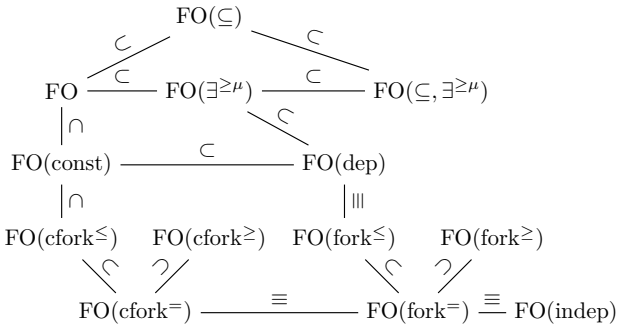


Figure 1 Overview of extensions of first-order logic by counting constructs over teams.

be the game from Proposition 19 such that $(\mathfrak{A}^*, X) \models \forall\overline{x}\forall\overline{\mu}(X\overline{x\mu} \to \varphi(\overline{x}, \overline{\mu}))$ if, and only if, X^* is an I-trap in $\mathcal{T}^{\#}(\mathfrak{A}, \varphi)$. Further, let $J(\varphi)$ be the interpretation, with coordinate map h, which, for every structure $\mathfrak{A}^*$, interprets $\mathcal{T}^{\#}(\mathfrak{A}^*, \varphi)$ in $\mathfrak{A}^*$.

Take now a formula $\mathrm{itrap}(x)$ of $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$ such that, for all threshold games $\mathcal{T}$, we have that $\mathcal{T} \models_Z \mathrm{itrap}(x)$ if, and only if, Z defines an I-trap in $\mathcal{T}$ (see Sect. 5.2). By the Interpretation Lemma we get a formula $\mathrm{itrap}^{J(\varphi)}(\lambda, \overline{v})$, also from $\mathrm{FO}(\subseteq, \exists^{\geq\mu})$, which is true in $\mathfrak{A}^*$ precisely for those teams $Y = h^{-1}(Z)$ where Z defines an I-trap of $\mathcal{T}^{\#}(\mathfrak{A}^*, \varphi)$. Specifically, we can write $\overline{v} = (\overline{x\mathrm{x}'}, \overline{\mu\mu'})$, so that for the team that defines $X^* = \{(\varphi, s) : s(\overline{x}, \overline{\mu}) \in X\}$ we get that $Y(X) = h^{-1}(X^*)$ is the set of all assignments $(\lambda, \overline{x\mathrm{x}'}, \overline{\mu\mu'}) \mapsto (0, \overline{a\mathrm{c}}, \overline{m\mathrm{0}})$ for some constant c and with $(\overline{a}, \overline{m}) \in X$. We now set

$$\psi(\overline{x}, \overline{\mu}) := \exists\lambda\exists\overline{x}'\exists\overline{\mu}'(\lambda = 0 \wedge \overline{x}' = \overline{c} \wedge \overline{\mu}' = \overline{0} \wedge \mathrm{itrap}^{J(\varphi)}(\lambda, \overline{x\mathrm{x}'}, \overline{\mu\mu})).$$

This implies that $\mathfrak{A}^* \models_X \psi(\overline{x}, \overline{\mu}) \iff \mathfrak{A}^* \models_{Y(X)} \mathrm{itrap}^{J(\varphi)}(\lambda, \overline{x\mathrm{x}'}, \overline{\mu\mu'})$. Putting everything together, we have

$$\begin{aligned}(\mathfrak{A}^*, X) \models \forall\overline{x\mu}(X\overline{x\mu} \to \varphi(\overline{x}, \overline{\mu})) &\iff X^* \text{ is an } I\text{-trap in } \mathcal{T}^{\#}(\mathfrak{A}^*, \varphi)\\ &\iff \mathcal{T}^{\#}(\mathfrak{A}^*, \varphi) \models_{X^*} \mathrm{itrap}(x) \iff \mathfrak{A}^* \models_{Y(X)} \mathrm{itrap}^{J(\varphi)}(\lambda, \overline{x\mathrm{x}'}, \overline{\mu\mu'})\\ &\iff \mathfrak{A}^* \models_X \psi(\overline{x}, \overline{\mu}).\end{aligned}$$

◀

6 Conclusion

We have explored two main variants for counting constructs in team semantics: forking atoms and counting quantifiers. We have seen that forking atoms are rather powerful, and indeed, dependence and independence logic are equivalent to variants of logics with forking. Counting quantifiers, on the other side, are most interesting in the context of inclusion logic, and we have shown that the extension of inclusion logic by counting quantifiers captures, in a precise sense, fixed-point logic with counting, which provides further interesting connections between team semantics and descriptive complexity theory. To establish the relationship between these two logics we have introduced a new variant of model checking games, threshold safety games, and interpretation arguments for studying them, which we believe to be of intrinsic interest beyond the results of this paper.

An open problem in this context concerns the relationship of different logics that all share the property of closure under unions. Inclusion atoms are closed under arbitrary unions of teams, and the same is true for forking atoms of type $\overline{v} \lhd^{\geq\mu} \overline{w}$. Further, counting quantifiers $\exists^{\geq\mu}$ preserve closure under union. By using different combinations of inclusion atoms, forking

atoms, and counting quantifiers we thus obtain a number of logics, all of which are closed under unions of teams, but whose relationship and expressive power is not really clear. We conjecture in particular that inclusion logic and FO(fork$^{\geq}$) are incomparable.

We remark that, due to the second-order features of team semantics, there is a rich variety of other potential counting constructs related to teams (some of which take the logics beyond Σ^1_1), and the research that we presented here is just a starting point. For instance, one may count the number of subteams of the given team that satisfy a given property, or develop an approach via generalized quantifiers. In [2] a kind of second-order majority quantifier has been considered, which counts witness functions for extending a given team by a new variable, and leads to a logic that captures the polynomial counting hierarchy.

Finally, one of the most important current challenges in the field of logics of dependence and independence is the systematic development of multi-team semantics, where assignments in teams may occur with multiplicities. This is fundamental for instance for reasoning about statistical (rather than logical) dependence and independence. It is obvious that the framework of two-sorted structures and counting constructs may be relevant for this project.

References

1 A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.
2 A. Durand, J. Ebbing, J. Kontinen, and H. Vollmer. Dependence logic with a majority quantifier. In *Proceedings of FSTTCS 2011*, pages 252–263, 2011.
3 F. Engström. Generalized quantifiers in dependence logic. *Journal of Logic, Language, and Information*, 2012.
4 P. Galliani. Inclusion and exclusion in team semantics – on some logics of imperfect information. *Annals of Pure and Applied Logic*, 163:68–84, 2012.
5 P. Galliani and L. Hella. Inclusion logic and fixed-point logic. In *Computer Science Logic 2013*, pages 281–295, 2013.
6 E. Grädel. Games for inclusion logic and fixed-point logic. In S. Abramsky et al., editor, *Dependence Logic. Theory and Applications*. Birkhäuser, 2016.
7 E. Grädel and J. Väänänen. Dependence and independence. *Studia Logica*, 101(2):399–410, 2013.
8 Y. Gurevich and S. Shelah. Fixed-point extensions of first-order logic. *Annals of Pure and Applied Logic*, 32:265–280, 1986.
9 W. Hodges. *Model Theory*. Cambridge University Press, 1993.
10 W. Hodges. Compositional semantics for a logic of imperfect information. *Logic Journal of IGPL*, 5:539–563, 1997.
11 N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.
12 J. Kontinen and J. Väänänen. On definability in dependence logic. *Journal of Logic, Language, and Information*, 18:317–241, 2009.
13 S. Kreutzer. Expressive equivalence of least and inflationary fixed point logic. *Annals of Pure and Applied Logic*, 130:61–78, 2004.
14 R. Rönnholm. Capturing k-ary existential second-order logic with k-ary inclusion-exclusion logic. arXiv:1502.05632v2, 2015.
15 J. Väänänen. *Dependence Logic*. Cambridge University Press, 2007.

The Logical Strength of Büchi's Decidability Theorem

Leszek Aleksander Kołodziejczyk[*1], **Henryk Michalewski**[†2], **Cécilia Pradic**[‡3], **and Michał Skrzypczak**[§4]

1 **University of Warsaw, Institute of Mathematics, Warsaw, Poland**
`l.kolodziejczyk@mimuw.edu.pl`
2 **University of Warsaw, Institute of Mathematics, Warsaw, Poland**
`h.michalewski@mimuw.edu.pl`
3 **ENS Lyon, Lyon, France**
4 **University of Warsaw, Institute of Informatics, Warsaw, Poland**
`m.skrzypczak@mimuw.edu.pl`

Abstract

We study the strength of axioms needed to prove various results related to automata on infinite words and Büchi's theorem on the decidability of the MSO theory of $(\mathbb{N}, \leq)$. We prove that the following are equivalent over the weak second-order arithmetic theory RCA_0:

1. Büchi's complementation theorem for nondeterministic automata on infinite words,
2. the decidability of the depth-n fragment of the MSO theory of $(\mathbb{N}, \leq)$, for each $n \geq 5$,
3. the induction scheme for Σ^0_2 formulae of arithmetic.

Moreover, each of (1)–(3) is equivalent to the additive version of Ramsey's Theorem for pairs, often used in proofs of (1); each of (1)-(3) implies McNaughton's determinisation theorem for automata on infinite words; and each of (1)-(3) implies the "bounded-width" version of König's Lemma, often used in proofs of McNaughton's theorem.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases nondeterministic automata, monadic second-order logic, Büchi's theorem, additive Ramsey's theorem, reverse mathematics

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.36

1 Introduction

Büchi's theorem [3] states that the monadic second-order theory of $(\mathbb{N}, \leq)$ is decidable. This is one of the fundamental results on the decidability of logical theories, and no less fundamental are the methods developed in order to prove it.

Typical proofs of Büchi's theorem make use of automata on infinite words. Büchi's original argument involved obtaining a complementation theorem for nondeterministic word automata: for each such automaton $\mathcal{A}$, there is another automaton $\mathcal{B}$ which accepts a given word exactly if $\mathcal{A}$ does not. Thanks to the complementation theorem, an MSO formula can be inductively translated into an equivalent nondeterministic automaton. At that point,

* Partially supported by Polish National Science Centre grant no. 2013/09/B/ST1/04390.
† Partially supported by Polish National Science Centre grant no. 2014-13/B/ST6/03595.
‡ Partially supported by Polish National Science Centre grant no. 2014-13/B/ST6/03595.
§ Partially supported by Polish National Science Centre grant no. 2014-13/B/ST6/03595.

© Leszek Kołodziejczyk, Henryk Michalewski, Cécilia Pradic, and Michał Skrzypczak;
licensed under Creative Commons License CC-BY

checking satisfiability of the formula becomes a matter of elementary combinatorics. Another approach to decidability of MSO was presented by Shelah in [16]. Shelah's "composition method" is automata-free, but is similar to Büchi's proof in one important respect: both use a restricted form of Ramsey's Theorem.

McNaughton [12] showed that an infinite word automaton can be determinised, though at the cost of allowing automata with a more general acceptance condition than Büchi's. Since deterministic automata are easy to complement, this again gives the translation of formulae to automata and thus decidability of MSO. To the best of our knowledge all determinisation proofs known from the literature rely on either a restricted form of Ramsey's Theorem or a restricted form of König's Lemma.

It is natural to ask how the various proofs of Büchi's theorem and related results compare to one another. For instance, is determinisation of word automata an "essentially stronger" result than complementation? Also, is the use of mildly nonconstructive principles à la Ramsey or König unavoidable?

A convenient framework for studying questions of this sort is provided by the programme of *reverse mathematics* [17]. The idea is to compare various theorems as formalised in the very expressive language of an axiomatic theory known as *second-order arithmetic*. Typical subtheories of second-order arithmetic are axiomatised by principles asserting the existence of more or less complicated sets of natural numbers. An important example is the relatively weak theory RCA_0, which guarantees only the existence of decidable sets. RCA_0 can formalise a significant amount of everyday mathematics and prove the termination of any primitive recursive algorithm, but it is unable to prove the existence of noncomputable objects such as the homogeneous sets postulated by Ramsey's Theorem or the infinite branches postulated by König's Lemma. Sometimes it is possible to show that two theorems not provable in RCA_0 are provably equivalent in it, and thus neither theorem is logically stronger than the other in the sense of requiring more abstract or less constructive sets. It is also often the case that a set existence principle used to derive some theorem is actually implied by the theorem over RCA_0. This serves as evidence that the principle is in fact necessary to prove the theorem.

In this paper, we carry out a reverse-mathematical study of the results around Büchi's theorem. We have two main aims in mind. One is to compare complementation, determinisation and decidability of MSO in terms of logical strength. The other aim is to clarify the role of Ramsey's Theorem and König's Lemma in proofs of Büchi's theorem and the related facts about automata. This seems interesting in light of the fact that the usual formulation of Ramsey's Theorem for pairs and the so-called Weak König's Lemma (the form of König's Lemma most commonly needed in practice) are known to be incomparable over RCA_0 [6, 11].

Our findings are as follows: firstly, determinisation of infinite word automata is no stronger than complementation, at least in the sense of implication over RCA_0. Secondly, decidability of MSO over $(\mathbb{N}, \leq)$ implies both complementation and determinisation. Finally, the use of Ramsey- or König-like principles in proofs of Büchi's theorem is mostly spurious in the sense that the versions that are actually needed follow from a very limited set-existence principle, namely mathematical induction for properties expressed by Σ^0_2 formulae. More precisely, we prove:

▶ **Theorem 1.** *Over* RCA_0*, the following statements are equivalent:*

1. *the principle of mathematical induction for* Σ^0_2 *formulae (denoted* Σ^0_2-IND*),*
2. *the Additive Ramsey Theorem (see Definition 2),*
3. *complementation for Büchi automata: there exists an algorithm which for each nondeterministic Büchi automaton* $\mathcal{A}$ *outputs a Büchi automaton* $\mathcal{B}$ *such that for every infinite word* α*,* $\mathcal{B}$ *accepts* α *exactly if* $\mathcal{A}$ *does not accept* α*,*

4. *the decidability of the depth-n fragment of the MSO theory of $(\mathbb{N}, \leq)$ (where $n \geq 5$ is a natural number)*[1].

Furthermore, each of 1.–4. implies:

5. *determinisation of Büchi automata: there exists an algorithm which for each nondeterministic Büchi automaton $\mathcal{A}$ outputs a deterministic Rabin automaton $\mathcal{B}$ such that for every infinite word α, $\mathcal{B}$ accepts α exactly if $\mathcal{A}$ accepts α.*

We also give a precise statement of the bounded-width form of König's Lemma often used in proofs of Item 5., and show that it is implied by each of 1.–4. Interestingly, it is not clear if 5. implies 1.–4. over RCA_0: standard arguments used to complement deterministic automata with acceptance conditions other than Büchi seem to involve Σ^0_2-IND.

It follows from our results that Büchi's theorem is unprovable in RCA_0, but only barely: it is true in computable mathematics, in the sense that the theorem remains valid if all the set quantifiers are restricted to range over (exactly) the decidable subsets of $\mathbb{N}$. This is in stark contrast to the behaviour of Rabin's theorem on the decidability of MSO on the infinite binary tree, which is known to require the existence of extremely complicated noncomputable sets [9]. Also Additive Ramsey's Theorem and Bounded-width König's Lemma are true in computable mathematics—quite unlike more general forms of Ramsey's Theorem for pairs and König's Lemma [7, 10].

To prove the implication $(4 \to 1)$ of Theorem 1, we come up with a family of MSO sentences for which truth in $(\mathbb{N}, \leq)$ is undecidable if Σ^0_2-IND fails. The other implications are proved by formalising more or less standard arguments from automata theory. In some cases this is routine, but especially the proof of $(1 \to 5)$ is quite delicate: we have to check not only that Σ^0_2-IND implies Bounded-width König's Lemma, but also that constructing the objects to which we apply the lemma is within the means of $\mathsf{RCA}_0 + \Sigma^0_2$-IND.

The structure of the paper is as follows. Sections 2 and 3 discuss the necessary background on reverse mathematics, automata, and MSO. We prove $(1 \to 2)$ in Section 4, $(2 \to 3)$ in Section 5, $(3 \to 4)$ in Section 6, $(4 \to 1)$ in Section 7. Section 8 contains a proof that Σ^0_2-IND implies Bounded-width König's Lemma, which is then applied to prove $(1 \to 5)$ in Section 9.

2 Background on reverse mathematics

Reverse mathematics [17] is a framework for studying the strength of axioms needed to prove theorems of countable mathematics, that is, the part of mathematics concerned with objects that can be represented using no more than countably many bits of information. This encompasses the vast majority of the mathematics needed in computer science.

The basic idea of reverse mathematics is to analyse mathematical theorems in terms of subsystems of a strong axiomatic theory known as second-order arithmetic. The two-sorted language of second-order arithmetic, L_2, contains *first-order* variables $x, y, z, \ldots$ (or $i, j, k, \ldots$), intended to range over natural numbers, and *second-order* variables $X, Y, Z, \ldots$, intended to range over sets of natural numbers. L_2 includes the usual arithmetic functions and relations $+, \cdot, \leq, 0, 1$ on the first-order sort, and the $\in$ relation which has one first-order and one second-order argument. The intended model of Z_2 is $(\omega, \mathcal{P}(\omega))$.

Notational convention. From this point onwards, we will use the letter $\mathbb{N}$ to denote the natural numbers as formalised in second-order arithmetic, that is, the domain of the first-

[1] The restriction to fixed-depth fragments is a technicality related to undefinability of truth. This is explained in more detail in Section 3.

order sort. On the other hand, the symbol ω will stand for the concrete, or standard, natural numbers. For instance, given a theory T and a formula $\varphi(x)$, "T proves $\varphi(n)$ for all $n \in \omega$" will mean "$\mathrm{T} \vdash \varphi(0), \mathrm{T} \vdash \varphi(1), \ldots$", which does not have to imply $\mathrm{T} \vdash \forall x \in \mathbb{N}.\, \varphi(x)$.

Full second-order arithmetic, Z_2, has axioms of three types: (i) axioms stating that the first-order sort is the non-negative part of a discretely ordered ring; (ii) comprehension axioms, or sentences of the form

$$\forall \bar{Y}\, \forall \bar{y}\, \exists X\, \forall x \left(x \in X \Leftrightarrow \varphi(x, \bar{Y}, \bar{y})\right),$$

where φ is an arbitrary formula of L_2 not containing the variable X; (iii) the induction axiom,

$$\forall X \left[0 \in X \wedge \forall x \left(x \in X \Rightarrow x + 1 \in X\right) \Rightarrow \forall x.\ x \in X\right].$$

The language L_2 is very expressive, as the first-order sort can be used to encode arbitrary finite objects and the second-order sort can encode even such objects as complete separable metric spaces, continuous functions between them, and Borel sets within them (cf. [17, Chapters II.5, II.6, and V.3]). Moreover, the theory Z_2 is powerful enough to prove almost all theorems from a typical undergraduate course that are expressible in L_2. In fact, the basic observation underlying reverse mathematics [17] is that many important theorems are equivalent to various fragments of Z_2, where the equivalence is proved in some specific weaker fragment, referred to as the *base theory*.

Quantifier hierarchies. Typical fragments of Z_2 are defined in terms of well-known quantifier hierarchies whose definitions we now recall. A formula is Σ^0_n if it has the form $\exists \bar{x}_1\, \forall \bar{x}_2\, \ldots\, \mathsf{Q} \bar{x}_n.\ \psi$, where the $\bar{x}_i$'s are blocks of first-order variables, the shape of Q depends on the parity of n, and ψ is Δ^0_0, i.e. contains only bounded first-order quantifiers. A formula is Π^0_n if it is the negation of a Σ^0_n formula. A formula is *arithmetical* if it contains only first-order quantifiers (second-order parameters are allowed).

A formula is Σ^1_n if it has the form $\exists \bar{X}_1\, \forall \bar{X}_2\, \ldots\, \mathsf{Q} \bar{X}_n.\ \psi$, where the $\bar{X}_i$'s are blocks of first-order variables, the shape of Q depends on the parity of n, and ψ is arithmetical. A formula is Π^1_n if it is the negation of a Σ^1_n formula.

In practice, we say that a formula is Σ^i_n/Π^i_n if it equivalent to a Σ^i_n/Π^i_n formula in the axiomatic theory we are working in at a given point.

Definition of RCA_0. The usual base theory in reverse mathematics is RCA_0, which guarantees only the existence of decidable sets. RCA_0 is defined by restricting the comprehension scheme to Δ^0_1-comprehension, which takes the form:

$$\forall \bar{Y}\, \forall \bar{y}\ \left[\forall x \left(\varphi(x, \bar{Y}, \bar{y}) \Leftrightarrow \neg \psi(x, \bar{Y}, \bar{y})\right) \Rightarrow \exists X\, \forall x \left(x \in X \Leftrightarrow \varphi(x, \bar{Y}, \bar{y})\right)\right],$$

where both φ and ψ are Σ^0_1. For technical reasons, it is necessary to strengthen the induction axiom to Σ^0_1-IND, that is, the scheme Σ^0_1-IND consisting of the sentences

$$\forall \bar{Y}\, \forall \bar{y}\ \left[\varphi(0, \bar{Y}, \bar{y}) \wedge \forall x\ \left(\varphi(x, \bar{Y}, \bar{y}) \Rightarrow \varphi(x + 1, \bar{Y}, \bar{y})\right) \Rightarrow \forall x.\ \varphi(x, \bar{Y}, \bar{y})\right]$$

for φ in Σ^0_1. Σ^0_1-IND makes it possible to define sequences by primitive recursion (cf. [17, Theorem II.3.4]): given some x_0 and a function $f\colon \mathbb{N} \to \mathbb{N}$, RCA_0 proves that there is a unique sequence $(x_i)_{i \in \mathbb{N}}$ such that $x_{i+1} = f(x_i)$ for each i.

RCA_0 has a unique minimal model in the sense of embeddability. This minimal model is (ω, Dec), where Dec is the family of decidable subsets of ω.

The Σ^0_n-IND scheme. In this paper we study an extension of RCA_0 obtained by strengthening the induction scheme to Σ^0_2 formulae. In general, for $n\in\omega$, the axiom scheme Σ^0_n-IND is defined just like Σ^0_1-IND, but with the induction formula φ in Σ^0_n rather than Σ^0_1. For each n, $\mathsf{RCA}_0+\Sigma^0_n$-$\mathsf{IND}$ is equivalent to $\mathsf{RCA}_0+\Pi^0_n$-IND, where the latter is defined in the natural way, as well as to the least number principle for Σ^0_n or Π^0_n formulae (cf. [17, Chapter II.3]).

Two important principles provable from Σ^0_n-IND are Σ^0_n-*collection*:

$$\forall \bar{Z}\,\forall \bar{z}\,\big[\forall x\leq t\,\exists y.\ \varphi(x,y,\bar{Z},\bar{z})\big]\Rightarrow \exists w\,\forall x\leq t\,\exists y\leq w.\ \varphi(x,y,\bar{Z},\bar{z}),$$

for φ in Σ^0_n, and *bounded* Σ^0_n-*comprehension*:

$$\forall \bar{Y}\,\forall \bar{y}\,\forall w\,\exists X\,\forall x\,(x\in X\Leftrightarrow x\leq w\wedge\varphi(x,\bar{Y},\bar{y})),$$

for φ in Σ^0_n.

For each n, the theory $\mathsf{RCA}_0+\Sigma^0_{n+1}$-$\mathsf{IND}$ is strictly stronger than $\mathsf{RCA}_0+\Sigma^0_n$-$\mathsf{IND}$ (cf. e.g. [4, Theorem IV.1.29]). However, note that the minimal model (ω,Dec) of RCA_0 satisfies $\mathsf{RCA}_0+\Sigma^0_n$-$\mathsf{IND}$ for all n, because an induction axiom is always true in a model with first-order universe ω.

Additive Ramsey's Theorem and Bounded-width König's Lemma. Two prominent extensions of RCA_0 are related to weak forms of important nonconstructive set existence principles: König's Lemma and Ramsey's Theorem.

Weak König's Lemma is the statement: "for every k, every infinite tree contained in $\{0,1,\dots,k\}^*$ has an infinite branch". The theory obtained by adding this statement to RCA_0 is known as WKL_0. This is the minimal theory supporting all sorts of "compactness arguments" in combinatorics, topology, analysis, and elsewhere (cf. [17, Chapter IV]).

The theory RT^2_2 extends RCA_0 by an axiom expressing Ramsey's Theorem for pairs and two colours[2]: for every 2-colouring of $[\mathbb{N}]^2$ there exists an infinite homogeneous set. $\mathsf{RT}^2_{<\infty}$ is defined similarly but allowing k-colourings for each $k\in\mathbb{N}$.

Both RT^2_2 and $\mathsf{RT}^2_{<\infty}$ are known to be incomparable with WKL_0 in the sense of implication over RCA_0 [6, 11]. WKL_0, RT^2_2, and RT^2_∞ are all false in the minimal model (ω,Dec) of RCA_0 [7, 10]. Much more on these theories can be found in [5].

In this paper, we study specific restricted versions of WKL_0 and $\mathsf{RT}^2_{<\infty}$ which play a role in proofs of Büchi's theorem. Recall that a *semigroup* is a set S with an associative operation $*\colon S\times S\to S$.

▶ **Definition 2** (Additive Ramsey Theorem)**.** The *Additive Ramsey Theorem* is the following statement: for every finite semigroup $(S,*)$ and every colouring $C\colon[\mathbb{N}]^2\to S$ such that for every $i<j<k$ we have $C(i,j)*C(j,k)=C(i,k)$, there exists an infinite homogeneous set $I\subseteq\mathbb{N}$. That is, there is a fixed color a such that for every $(i,j)\in[I]^2$, $C(i,j)=a$.

▶ **Definition 3** (Bounded-width König's Lemma)**.** *Bounded-width König's Lemma* is the following statement: for every finite set Q and every graph G whose vertices belong to $Q\times\mathbb{N}$ and whose edges are all of the form $((q,i),(q',i+1))$ for some $q,q'\in Q$, if there are arbitrarily long finite paths in G starting in some vertex $(q,0)$, then there is an infinite path in G starting in $(q,0)$.

[2] By $[X]^2$ we denote the set of unordered pairs of elements of X.

Notice that Bounded-width König's Lemma applied to a graph G is essentially the same as Weak König's Lemma applied to the tree obtained by the so-called unraveling of G (in particular, Bounded-width König's Lemma is provable in WKL_0). However, the graph formulation is more natural to express.

3 Background on MSO and Büchi automata

Büchi automata and MSO logic are equivalent formalisms for specifying properties of infinite words. In this section we formally introduce these concepts. If not stated otherwise, the formalisation presented here is carried out in RCA_0.

Infinite words. By Σ we denote a finite nonempty set called an *alphabet*. A *finite word* over Σ is a function $w\colon \{0,\dots,k-1\} \to \Sigma$; the *length* of w is k. The set of all finite words over Σ is denoted Σ^*. An *infinite word* over Σ is a function $\alpha\colon \mathbb{N} \to \Sigma$. We write $\alpha \in \Sigma^{\mathbb{N}}$ for "α is an infinite word over Σ".

Every infinite word can be treated as a relational structure with the universe $\mathbb{N}$ and predicates: the binary order predicate $\leq$ and a unary predicate a for every $a \in \Sigma$. The semantics of these predicates over a given infinite word α is natural, in particular $a(x)$ holds if $\alpha(x) = a$.

When working with automata and logic it is customary to define *languages*—sets of infinite words satisfying certain properties. However, from the point of view of second-order arithmetic a language is a "third-order object". Therefore, in this paper we avoid talking directly about languages. Instead, when we want to express some properties of languages, we explicitly quantify over infinite words with a given property.

Automata over infinite words. A (nondeterministic) Büchi automaton is a tuple $\mathcal{A} = \langle Q, \Sigma, q_{\mathrm{I}}, \delta, F\rangle$ where: Q is a finite set of *states*, Σ is an alphabet, $q_{\mathrm{I}} \in Q$ is an *initial state*, $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, and $F \subseteq Q$ is the set of *accepting states*. Given an infinite word $\alpha \in \Sigma^{\mathbb{N}}$, we say that $\rho \in Q^{\mathbb{N}}$ is a *run* of $\mathcal{A}$ over α if $\rho(0) = q_{\mathrm{I}}$ and for every $n \in \mathbb{N}$ we have $\big(\rho(n), \alpha(n), \rho(n+1)\big) \in \delta$. A run ρ is *accepting* if $\rho(n) \in F$ for infinitely many $n \in \mathbb{N}$. An automaton $\mathcal{A}$ *accepts* α if there exists an accepting run of $\mathcal{A}$ over α. An automaton is *deterministic* if for every $q \in Q$ and $a \in \Sigma$ there is at most one transition of the form $(q, a, q') \in \delta$. When the automaton is not clear from the context, we put it in the superscript, i.e. $Q^{\mathcal{A}}$ is the set of states of $\mathcal{A}$.

The possible transitions of a Büchi automaton over a particular letter $a \in \Sigma$ can be encoded as a *transition matrix* $M_a\colon Q \times Q \to \{0, 1, \star\}$, where $M_a(q, q') = 0$ if $(q, a, q') \notin \delta$, otherwise $M_a(q, q') = \star$ if $q \in F$, and otherwise $M_a(q, q') = 1$. Let $[Q]$ be the set of all such functions $M\colon Q \times Q \to \{0, 1, \star\}$.

Since deterministic Büchi automata are strictly weaker than general Büchi automata [14], one introduces the more flexible *Rabin acceptance condition* in order to determinise Büchi automata. A *Rabin automaton* is a tuple $\mathcal{A} = \langle Q, \Sigma, q_{\mathrm{I}}, \delta, (E_i, F_i)_{i=1}^k\rangle$ as in the case of Büchi automata, where $E_i, F_i \subseteq Q$ for $i = 1, \dots, k$. A run $\rho \in Q^{\mathbb{N}}$ of $\mathcal{A}$ is *accepting* if and only if for some $i \in \{1, \dots, k\}$ each state in E_i appears only finitely many times in ρ and some state in F_i appears infinitely many times in ρ.

In general (i.e. in Z_2) Rabin automata can easily be complemented into so-called *Streett automata*, and both classes can be transformed into nondeterministic Büchi automata. However, the transformations into Büchi automata require more than RCA_0. For Streett automata, Σ^0_2-IND seems necessary. For Rabin, we need the Büchi automaton to guess that

no state from a given set of states will reappear in the run under consideration. To prove that such a construction is correct one needs Σ^0_2-collection—within RCA_0 the fact that in a given run ρ each state $q \in E$ appears only finitely many times does not imply a global bound after which no state from E reappears. That is the essential reason why it is not clear whether Item 5. of Theorem 1 implies the other items in RCA_0.

Monadic Second-Order logic. Monadic second-order logic (MSO) is an extension of first-order logic. MSO logic allows: boolean connectives $\neg$, $\vee$, $\wedge$; the first-order quantifiers $\exists x$ and $\forall x$; and the monadic second-order quantifiers $\exists X$ and $\forall X$, where the variable X ranges over subsets of the universe. Apart from predicates from the signature of a given structure, the logic admits the binary predicate $x \in X$ with the natural semantics.

Definition of truth for MSO over $\mathbb{N}$. In order to state our theorems involving decidability of the MSO theory of $(\mathbb{N}, \leq)$, we need to formulate the semantics of monadic second-order logic within RCA_0. This involves a coding of formulae $\phi \mapsto \lceil \phi \rceil$; we identify a formula with its code. However, in second-order arithmetic there is no canonical definition of truth in an infinite structure which would work for all of MSO. Moreover, by Tarski's theorem on the undefinability of truth, for some infinite structures there is no such definition at all. In particular, it is not at all clear how to state the decidability of $\mathrm{MSO}(\mathbb{N}, \leq)$ as a single sentence.

On the other hand, already RCA_0 is able to express a truth definition for the *depth-n fragment of MSO*, for each $n \in \omega$. Here the depth of a formula is calculated as the largest number of alternating blocks of $\wedge/\forall$'s and $\vee/\exists$'s appearing on a branch in the syntactic tree of the formula (assume that all negations are pushed inside using the De Morgan laws). Essentially, the truth definition needs one universal set quantifier for a block of $\wedge/\forall$'s and one existential set quantifier for a block of $\vee/\exists$'s[3].

So, what is possible is to provide formulae φ_n stating that the depth-n fragment of $\mathrm{MSO}(\mathbb{N}, \leq)$ is decidable. We show in Section 6 that every φ_n can be proved in RCA_0 assuming a complementation procedure for Büchi automata, and in Section 7 that φ_5 implies Σ^0_2-IND. As a corollary, we can observe that $\mathsf{RCA}_0 \vdash \varphi_5 \Rightarrow \varphi_n$ for every $n \in \omega$.

The Büchi decidability theorem. In [3] Büchi proved decidability of the theory $\mathrm{MSO}(\mathbb{N}, \leq)$. The following theorem captures as much of Büchi's result as can be naturally expressed in relatively weak theories of second-order arithmetic.

▶ **Theorem 4** (Büchi formalised). *There exists an effective procedure P such that for every fixed depth $n \in \omega$ the following is provable in $\mathsf{RCA}_0 + \Sigma^0_2$-$\mathsf{IND}$. For every statement ϕ of MSO over an alphabet Σ such that the depth of ϕ is at most n, the procedure $P(\phi)$ produces a nondeterministic Büchi automaton $\mathcal{A}$ over Σ such that for every infinite word $\alpha \in \Sigma^{\mathbb{N}}$, this word satisfies ϕ if and only if $\mathcal{A}$ accepts α. Moreover, it is decidable if a given nondeterministic Büchi automaton accepts any infinite word.*

We discuss some issues related to formalising the inductive proof of Büchi's theorem in Section 6. The crucial step concerns complementation of automata, which is used to treat negations of subformulae in ϕ (or subformulae beginning with $\forall$, assuming the negations have been pushed inside).

[3] After slight modifications, the truth definition would still work if we allowed depth-n formulas to contain arbitrarily many alternations $\wedge$'s and $\vee$'s inside the scope of the last quantifier counted towards the depth.

4 Σ^0_2-IND implies Additive Ramsey

The aim of this section is to prove the following theorem.

▶ **Theorem 5.** *Over* RCA_0, Σ^0_2-IND *implies Additive Ramsey's Theorem (see Definition 2).*

The proof consists of two steps. First, we prove another weakening of Ramsey's Theorem.

▶ **Definition 6.** *Ordered Ramsey's Theorem* for pairs states that if $(P, \preceq)$ is a finite partial order and $C\colon [\mathbb{N}]^2 \to P$ is a colouring such that for every $i < j < k$ we have $C(i,j) \succeq C(i,k)$, then there exists an infinite homogeneous set $I \subseteq \mathbb{N}$, i.e. $C(i,j) = C(i',j')$ for all $(i,j),(i',j') \in [I]^2$.

Note that this statement follows immediately from the so-called *Stable Ramsey's Theorem* $\mathsf{SRT}^2_{<\infty}$ (cf. [5, Sections 6.4 and 6.8]), where the requirement on C is only that $C(i,\cdot)$ should stabilise for each i.

▶ **Lemma 7.** *Over* RCA_0, Σ^0_2-IND *proves Ordered Ramsey's Theorem.*

Proof. We call a colour $p \in P$ *recurring* if $\forall i\, \exists k > j > i.\ C(j,k) = p$. Notice that for each non-recurring colour p there exists i_p such that there is no occurrence of p to the right of i_p (i.e. no $k > j > i_p$ such that $C(j,k) = p$). By an application of Σ^0_2-collection we obtain some i_0 such that for every non-recurring colour p and every $k > j > i_0$ we have $C(j,k) \neq p$. In particular, there is a recurring colour. Moreover, being a recurring colour is a Π^0_2 property, so by Σ^0_2-IND we can find a $\preceq$-minimal recurring colour p_0.

We now define a sequence $(u_i, v_i)_{i\in\mathbb{N}}$ by primitive recursion on i. Let (u_0, v_0) be some pair such that $i_0 < u_0 < v_0$ and $c(u_0, v_0) = p_0$. Now assume that $u_0 < v_0 \leq u_1 < v_1 \ldots \leq u_i < v_i$ have been defined, $\{u_0, \ldots, u_i\}$ is homogeneous with colour p_0, and $C(u_i, v_i) = p_0$. Let (u_{i+1}, v_{i+1}) be the smallest pair such $v_i \leq u_{i+1} < v_{i+1}$ and $C(u_{i+1}, v_{i+1}) = p_0$. Such a pair exists because p_0 is recurring. We know that $C(u_i, u_{i+1}) = p_0$, since on the one hand $C(u_i, u_{i+1}) \preceq C(u_i, v_i) = p_0$, and on the other hand $u_i > i_0$ and thus $C(u_i, u_{i+1})$ is a recurring colour, so it cannot be $\preceq$-strictly smaller than p_0. Similarly, for $j < i$ we know that $C(u_j, u_{i+1}) = p_0$ because $C(u_j, u_{i+1}) \preceq p_0$ and $u_j > i_0$. Therefore, the set $\{u_i \mid i \in \mathbb{N}\}$ is homogeneous for C. ◀

Before proceeding to prove the additive version of the theorem, we recall a few basic facts about finite semigroups we shall use in our proof. The facts are proved by elementary combinatorial arguments which readily formalise in RCA_0. The proofs can be found for instance in [14].

▶ **Definition 8.** Green preorders over a semigroup S are defined as follows

- $s \leq_{\mathcal{R}} t$ if and only if $s = t$ or $s \in t * S = \{t * a \mid a \in S\}$,
- $s \leq_{\mathcal{L}} t$ if and only if $s = t$ or $s \in S * t = \{a * t \mid a \in S\}$,
- $s \leq_{\mathcal{H}} t$ if and only if $s \leq_{\mathcal{R}} t$, and $s \leq_{\mathcal{L}} t$.

The associated equivalence relations are written $\mathcal{R}$, $\mathcal{L}$, $\mathcal{H}$; their equivalence classes are called respectively $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{H}$-classes.

▶ **Lemma 9.** *For every finite semigroup* S *and* $s, t \in S$, $s \leq_{\mathcal{L}} t$ *and* $s\ \mathcal{R}\ t$ *implies* $s\ \mathcal{H}\ t$.

▶ **Lemma 10** ([14, Proposition 2.4])**.** *If* $(S, *)$ *is a finite semigroup,* $H \subseteq S$ *an* $\mathcal{H}$*-class, and some* $a, b \in H$ *satisfy* $a * b \in H$ *then for some* $e \in H$ *we know that* $(H, *, e)$ *is a group.*

Now we can prove our main statement.

Proof of Theorem 5. Let a colouring C take values in the finite semigroup $(S, *)$ and satisfy the additivity condition of Definition 2. For every position i and every $k \geq j > i$, let us observe that $C(i,k) \leq_{\mathcal{R}} C(i,j)$. Let r be the function mapping every element of S to its $\mathcal{R}$-class. The function $r \circ C$ is an ordered colouring; let us use Lemma 7 to obtain a homogeneous sequence $(u_i)_{i\in\mathbb{N}}$ for $r \circ C$.

Since S is finite, we can use Σ^0_2-collection to prove that there is some colour a such that $C(u_0, u_i) = a$ for infinitely many i. This lets us take a subsequence $(v_i)_{i\geq 0}$ of $(u_i)_{i\geq 0}$ such that $C(v_0, v_i) = a$ for each i.

We now know that $a = a * C(v_i, v_j)$ for every $0 < i < j$. In particular, $a \leq_{\mathcal{L}} C(v_i, v_j)$ by the definition of $\leq_{\mathcal{L}}$. Since a and $C(v_i, v_j)$ are $\mathcal{R}$-equivalent, Lemma 9 implies that $C(v_i, v_j)\ \mathcal{H}\ a$. Let H be the $\mathcal{H}$-class of a. Since $a * C(v_i, v_j) = a \in H$ we know by Lemma 10 that $(H, *, e)$ is a group for some $e \in H$. Using this group structure and the equation $a = a * C(v_i, v_j)$ we obtain that $C(v_i, v_j) = e$. Hence, $\{v_{i+1} \mid i \in \mathbb{N}\}$ is a homogeneous set for C with the colour e. ◀

We will now sketch the opposite implication, as stated by the following lemma. It follows from the other implications of Theorem 1, thus the reasoning presented here is not needed to obtain the theorem. However, we decided to include it, as the argument is very straightforward and avoids the use of automata and logic.

▶ **Lemma 11.** *Over* RCA_0*, Additive Ramsey's Theorem implies* Σ^0_2-IND.

Proof sketch. By the construction from Section 7, a failure of Σ^0_2-IND gives us $a \in \mathbb{N}$ and an infinite word $\alpha \in \{0, \ldots, a+1\}^{\mathbb{N}}$ such that there is no highest letter i that appears infinitely many times in α. Fix such a word α and consider the colouring with values in $\{0, \ldots, a+1\}$ defined for $i < j$ as follows:

$$C(i,j) = \max\{\alpha(k) \mid i \leq k < j\}.$$

Clearly, C is an additive colouring of $[\mathbb{N}]^2$ by elements of the semigroup $(\{0, \ldots, a+1\}, \max)$. Apply Additive Ramsey's Theorem to obtain an infinite homogeneous set $I \subseteq \mathbb{N}$ for C. Assume that $i \in \{0, \ldots, a+1\}$ is the colour of I. By the definition of C, i is the highest colour that appears infinitely many times in α. ◀

In the full version of the paper, we additionally provide a direct proof that Ordered Ramsey's Theorem implies Σ^0_2-IND.

5 Additive Ramsey implies complementation

In this section, we sketch a proof of the following theorem.

▶ **Theorem 12.** *Over* RCA_0*, the Additive Ramsey Theorem (see Definition 2) implies the following complementation result: there exists an algorithm which, given a Büchi automaton* $\mathcal{A}$ *over an alphabet* Σ*, outputs a Büchi automaton* $\mathcal{B}$ *over the same alphabet such that for every* $\alpha \in \Sigma^{\mathbb{N}}$ *we have that* $\mathcal{A}$ *accepts* α *if and only if* $\mathcal{B}$ *does not accept* α*.*

The proof of this theorem follows the standard construction of the automaton $\mathcal{B}$ [3]: the states of $\mathcal{B}$ are based on transition matrices of $\mathcal{A}$ (see Section 3). The automaton $\mathcal{B}$ guesses a Ramseyan decomposition of the given infinite word α with respect to a certain homomorphism into $[Q]$; and then verifies that the decomposition witnesses that there cannot be any accepting run of $\mathcal{A}$ over α. A complete proof of the theorem will be given in the full version of the paper.

6 Complementation implies decidability

▶ **Theorem 13.** *For any $n \in \omega$, the following is provable in* RCA_0*: if there exists an algorithm for complementing Büchi automata, then there exists an algorithm which, given an MSO formula ϕ of depth at most n, outputs an automaton $\mathcal{A}_\phi$ such that for every word ν, ν satisfies the formula ϕ if and only if ν is accepted by $\mathcal{A}_\phi$. As a consequence, the depth-n fragment of MSO$(\mathbb{N}, \leq)$ is decidable.*

A detailed proof of the theorem will be given in the full version of the paper. Te argument is along the lines of the standard inductive construction of an automaton $\mathcal{A}_\phi$ that simulates the behaviour of ϕ. Let us recall that in RCA_0 we only have truth definitions for fixed-depth MSO formulae. Additionally, each such truth definition is not a Σ^0_1 formula (it is not even arithmetical, as it quantifies over infinite words). Therefore, in RCA_0 we cannot perform any induction involving the truth definition. This fact has two consequences:

1. in the above theorem, the implication from complementation to decidability is stated for all $n \in \omega$ separately and its proof is obtained via an external induction over n,
2. our construction of $\mathcal{A}_\phi$ needs to work in a fixed number of steps (depending on n), no iterative procedure can be involved. In particular, we need to simulate whole blocks of quantifiers or connectives at once.

To complete the proof of the theorem, we verify in RCA_0 that the emptiness problem is decidable for Büchi automata, as expressed by the following lemma.

▶ **Lemma 14.** *Provably in* RCA_0*, it is decidable if, given a nondeterministic Büchi automaton $\mathcal{A}$, there exists an infinite word accepted by $\mathcal{A}$.*

7 Decidability implies Σ^0_2-IND

In this section we prove the following theorem.

▶ **Theorem 15.** *Over* RCA_0*, the decidability of the depth-5 fragment of MSO$(\mathbb{N}, \leq)$ implies* Σ^0_2-IND.

The rest of this section is devoted to a proof of this theorem. Consider a Π^0_2 formula (with parameters we keep implicit) $\phi(i) \equiv \forall x\, \exists y.\ \delta(i, x, y)$ and suppose it satisfies the premises of induction, i.e. $\phi(0)$ holds and $\forall i\, (\phi(i) \Rightarrow \phi(i+1))$. Take $a \in \mathbb{N}$. We want to show that $\phi(a)$ holds. For that we will use decidability of the depth-5 fragment of MSO$(\mathbb{N}, \leq)$ to prove using Σ^0_1-IND that a certain formula ψ_{a+1} is true in $(\mathbb{N}, \leq)$. We will construct a specific infinite word that encodes the semantics of $\phi(a)$ and use the fact that the word satisfies ψ_{a+1} to deduce that $\phi(a)$ holds.

For $k \in \mathbb{N}$ let ψ_k be the MSO formula stating "for every infinite word over the alphabet $\{0, \ldots, k\}$ there is a maximal letter $i \in \{0, \ldots, k\}$ occurring infinitely often". More formally, ψ_k is defined as follows.

$$\psi_k \equiv \quad \forall X_0\, \forall X_1\, \ldots \forall X_k \left[\forall x \left(\bigvee_{i \leq k} x \in X_i \wedge \bigwedge_{i < j \leq k} \neg (x \in X_i \wedge x \in X_j) \right) \Longrightarrow \right. \tag{1}$$

$$\left. \bigvee_{i \leq k} \left((\forall x\, \exists y \geq x.\ y \in X_i) \wedge \bigwedge_{i < j \leq k} (\exists x\, \forall y \geq x.\ y \notin X_j) \right) \right].$$

The formula ψ_k is an MSO formula of depth 5. By the assumption on decidability, the property that ψ_k belongs to the theory $\mathrm{MSO}(\mathbb{N},\leq)$ can be expressed by a Σ^0_1 formula of second-order arithmetic, $\Psi(k)$ (and, in fact, by a Π^0_1 formula as well). Clearly, in RCA_0 we can prove that ψ_0 belongs to $\mathrm{MSO}(\mathbb{N},\leq)$ and for every $i\in\mathbb{N}$, if ψ_i belongs to $\mathrm{MSO}(\mathbb{N},\leq)$, then ψ_{i+1} belongs to $\mathrm{MSO}(\mathbb{N},\leq)$. Therefore, by the assumption on Ψ, we know that $\Psi(0)$ holds and $\forall i\,(\Psi(i)\Rightarrow\Psi(i+1))$. Then, Σ^0_1-IND guarantees that $\Psi(a+1)$ is true and hence ψ_{a+1} belongs to $\mathrm{MSO}(\mathbb{N},\leq)$.

Now our aim is to construct a specific infinite word α over the alphabet $\{0,\dots,a+1\}$ in such a way to guarantee that Claim 16 below holds.

For $i\leq a$ and $w\in\mathbb{N}$ let $C(i,w)=\max\big\{v\leq w\mid \forall x{<}v\,\exists y{<}w.\ \delta(i,x,y)\big\}$.

Clearly the function $C(i,w)$ is computable. Assume a computable enumeration[4] for pairs $\langle\cdot,\cdot\rangle\colon\mathbb{N}^2\to\mathbb{N}$ that is monotone with respect to the coordinatewise order on $\mathbb{N}^2$. Define an infinite word

$$\alpha(n)=\begin{cases} i+1 & \text{if } n=\langle i,w\rangle,\ i\leq a,\text{ and } C(i,w)>\big|\{w'<w\mid\alpha(\langle i,w'\rangle)=i+1\}\big|,\\ 0 & \text{otherwise.}\end{cases}$$

Again, $\alpha(n)$ is computable so α can be defined by Δ^0_1-comprehension. We will prove in RCA_0 the following claim.

▶ **Claim 16.** *For every $i\leq a$ and $v\in\mathbb{N}$ the letter $i+1$ appears at least v times in α if and only if $\forall x{<}v\,\exists y.\ \delta(i,x,y)$. In particular, $i+1$ appears infinitely many times in α if and only if $\phi(i)$ holds.*

Proof. First assume that $\forall x{<}v\,\exists y.\ \delta(i,x,y)$ holds for some $i\leq a$ and $v\in\mathbb{N}$. By Σ^0_1-collection, there exists some w such that $\forall x{<}v\,\exists y{<}w.\ \delta(i,x,y)$. Let $k=\big|\{w'<w\mid\alpha(\langle i,w'\rangle)=i+1\}\big|$. If $k\geq v$ then we are done. Assume the contrary and notice that $C(i,w)\geq v$. This means that for $w'=w,w+1,\dots,w+v-k-1$ we have $\alpha(\langle i,w'\rangle)=i+1$ (we use Σ^0_1-IND to prove this). In total this gives us v positions of α that are labelled by $i+1$.

Now assume that there are at least v positions of α labelled by $i+1$. Let w_0 be the minimal position such that $\big|\{w'\leq w_0\mid\alpha(\langle i,w'\rangle)=i+1\}\big|=v$. In particular we know that $\alpha(\langle i,w_0\rangle)=i+1$ and $\big|\{w'<w_0\mid\alpha(\langle i,w'\rangle)=i+1\}\big|=v-1$. This means that $C(i,w_0)\geq v$. By the definition of $C(i,w)$, it follows that $\forall x{<}v\,\exists y.\ \delta(i,x,y)$ holds. ◀

Now we conclude the proof of Theorem 15. Since ψ_{a+1} holds, we know that its body holds for the sets X_i defined as $X_i=\{j\mid\alpha(j)=i\}$, $i=0,\dots,a+1$ (Δ^0_1-comprehension is used here). Clearly these sets form a partition of $\mathbb{N}$ and thus the formula ψ_{a+1} gives us an index $i\leq a+1$ such that i is the maximal letter that appears infinitely many times in α. Since $\phi(0)$ holds we know that $i>0$. If $i=a+1$ then by Claim 16 we obtain our thesis that $\phi(a)$ holds. Assume to the contrary that $i<a+1$. By Claim 16 it means that $\phi(i)$ and $\neg\phi(i+1)$ hold. This contradicts the assumption that $\forall i\,\big(\phi(i)\Rightarrow\phi(i+1)\big)$. Thus, a proof of $\phi(a)$ is concluded.

▶ Remark. The work of Sections 4–7 shows that the effectivity condition in Item 3. of Theorem 1 is not necessary to derive the other items in RCA_0. The bare statement that for every Büchi automaton there exists a complementing automaton already suffices.

The argument is as follows: assuming that each Büchi automaton can be complemented, the fixed-depth expressible property that a given word α does not satisfy the body of a

[4] $(n,k)\mapsto\frac{(n+k+1)(n+k)}{2}+k$ is one such map simple enough.

formula ψ_k as in (1) can be recognised by a Büchi automaton. By the proof of Lemma 14, if such an automaton accepts some infinite word, then it accepts an ultimately periodic infinite word. But this clearly shows that ψ_k is true for any k, thus proving Σ^0_2-IND and hence also the other items of Theorem 1.

8 Σ^0_2-IND implies Bounded-width König

$\mathsf{RCA}_0 + \Sigma^0_2$-IND is too weak to prove Weak König's Lemma (in fact, Σ^0_2-IND and WKL_0 are incomparable over RCA_0). However, it turns out that Σ^0_2-IND proves a restricted version of the lemma, where the "width" of the trees under consideration is globally bounded, in the sense that the subtree rooted in a vertex $\langle i_0, \dots, i_\ell \rangle \in \{0, \dots, k\}^*$ is completely determined by i_ℓ.

▶ **Theorem 17.** *Over* RCA_0, Σ^0_2-IND *implies Bounded-width König's Lemma (see Definition 3).*

Let us fix a graph G with vertices contained in $Q \times \mathbb{N}$ for some finite set Q. The usual way of proving König's Lemma starts by defining the subset G' of those vertices v of G for which the subgraph under v is infinite. Having defined G', we inductively pick any infinite path in G' and—assuming G does in fact contain arbitrarily long finite paths starting in $Q \times \{0\}$—we are guaranteed not to get stuck. The issue is whether we can obtain G' by Δ^0_1-comprehension.

A Π^0_1 definition of G' is provided by a standard trick used in the context of WKL_0. Notice that for every fixed n there can be at most $|Q|$ vertices of G of the form (q, n). Thus a vertex (q, n) is in G' if and only if it has the Π^0_1 property that for every $n' \geq n$ there exists a vertex (q', n') reachable from (q, n) by a path in G.

What remains is to give a Σ^0_1-definition of G'.

Consider two numbers $n < m$ and a vertex $v = (q, n)$ of G. We will say that v *dies before* m if there is no path in G from v that reaches a vertex of the form (q', m). For $i = 0, 1, \dots, |Q|$ we will say that i *vertices die infinitely many times* if

$$\forall k\, \exists n > k\, \exists m > n.\ \text{there are at least } i \text{ vertices of the form } (q, n) \text{ that die before } m.$$

Notice that the property of i that *i vertices die infinitely many times* is Π^0_2. Clearly if $i < i'$ and *i' vertices die infinitely many times* then *i vertices die infinitely many times*. By Σ^0_2-IND we can fix i_0 as the maximal i such that *i vertices die infinitely many times*. Notice that for each $i > i_0$ there exists $k(i)$ such that for every $m > n > k(i)$ there are fewer than i vertices of the form (q, n) that die before m. By Σ^0_2-collection, we can find a global bound k_0 such that $k_0 > k(i)$ for all $i > i_0$. This means that for $m > n > k_0$ we have at most i_0 vertices of the form (q, n) that die before m. Additionally, for infinitely many n there is $m > n$ such that exactly i_0 vertices of the form (q, n) die before m. The following claim shows how one can find a witness that the subgraph under a vertex v is infinite.

▶ **Claim 18.** *Assume that we are given $m > n > k_0$ and a vertex $v = (q, n)$ such that exactly i_0 vertices of the form (q', n) with $q' \neq q$ die before m. Then the subgraph under v is infinite.*

Proof. Assume to the contrary that for some $m' > m$ there is no vertex of the form (q', m') that can be reached from (q, n) by a path in G. It means that (q, n) dies before m'. Therefore, there are at least $i_0 + 1$ vertices of the form (q', n) that die before m'. This contradicts the way k_0 was chosen. ◀

Clearly, if for some $m > n$ and a vertex $v = (q, n)$ we know that v dies before m then the subgraph of G under v is finite.

We shall now use Claim 18 to give a Σ^0_1-definition of G'. We will say that $v = (q, n_0)$ belongs to G' if there exist $m > n > \max(k_0, n_0)$ and i_0 vertices of the form (q', n) such that all of them die before m and some other vertex of the form (q'', n) is reachable in G by a path from v. Clearly this is a Σ^0_1-definition. It remains to prove that it defines G'. First assume that v satisfies the above property and fix m, n, and (q'', n) as in the definition. By Claim 18 we know that the subgraph under (q'', n) is infinite. Since (q'', n) is reachable from v in G, this implies that also the subgraph under v is infinite and thus $v \in G'$. Now assume that $v = (q, n_0) \in G'$. By the choice of i_0 we know that there exist $m > n > \max(n_0, k_0)$ and exactly i_0 vertices of the form (q', n) that die before m. Since the subgraph under v is infinite, we know that some vertex of the form (p, m) is reachable from v in G. Notice that any path connecting v and (p, m) needs to contain a vertex of the form (q'', n). Clearly (q'', n) cannot be any of the i_0 vertices that die before m. Thus v satisfies the above condition.

▶ **Fact 19.** *If a vertex $(q, 0)$ of G satisfies the hypothesis of Bounded-width König's Lemma, then $(q, 0) \in G'$. Moreover, if $v = (q, n) \in G'$ then there exists $(q', n+1) \in G'$ such that there is an edge between (q, n) and $(q', n+1)$.*

Now, given $(q, 0) \in G'$, we can construct an infinite path in G' using Δ^0_1-comprehension. Fix any linear order on Q. Let $\pi(0)$ be $(q, 0)$. If $\pi(n)$ is defined let $\pi(n+1) = (q', n+1)$ for the minimal $q' \in Q$ satisfying: $(q', n+1) \in G'$ and there is an edge in G between $\pi(n)$ and $(q', n+1)$. Fact 19 implies that π is well-defined. By the construction π is an infinite path in G' and thus in G.

9 Σ^0_2-IND implies determinisation

In this section we will show the following theorem.

▶ **Theorem 20.** *Over* RCA_0, Σ^0_2-IND *implies the existence of an algorithm which, given a nondeterministic Büchi automaton $\mathcal{B}$ over an alphabet Σ, outputs an equivalent deterministic Rabin automaton $\mathcal{A}$—the alphabet of $\mathcal{A}$ is Σ and for every infinite word α over Σ, $\mathcal{A}$ accepts α if and only if $\mathcal{B}$ accepts α.*

The proof scheme presented here is based on a determinisation procedure proposed in [13] (see [1, 8] for similar arguments and a comparison of this determinisation method to the method of Safra). Our exposition follows lecture notes of M. Bojańczyk [2]. Although the general structure of the argument is standard, we need to take additional care to ensure that the reasoning can be conducted in RCA_0 using only Σ^0_2-IND.

The proof of Theorem 20 will be split into separate steps that will allow us to successively simplify the objects under consideration. To merge these steps we will use the notion of a deterministic transducer that transforms one infinite word into another.

▶ **Definition 21.** A transducer is a deterministic finite automaton, without accepting states, where each transition is additionally labelled by a letter from some *output alphabet*. More formally, a transducer with an input alphabet Σ and an output alphabet Γ is a tuple $\mathcal{T} = \langle Q, q_{\mathrm{I}}, \delta \rangle$ where $q_{\mathrm{I}} \in Q$ is an initial state and $\delta \colon Q \times \Sigma \to \Gamma \times Q$.

A transducer naturally defines a function $\mathcal{T} \colon \Sigma^{\mathbb{N}} \to \Gamma^{\mathbb{N}}$. Formally, such a function is a third-order object and thus not available in second-order arithmetic. However, given a word α, we can use Δ^0_1-comprehension to obtain the unique infinite word produced by $\mathcal{T}$ on input α. Whenever we write $\mathcal{T}(\alpha)$, we have this word in mind.

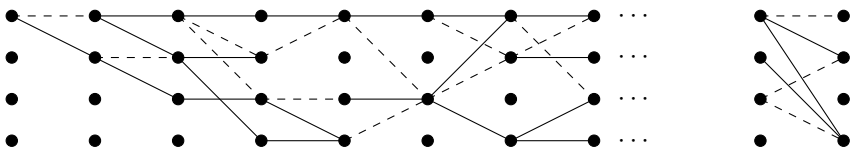


Figure 1 A Q-dag and a single letter from the alphabet $[Q]$. The accepting edges are represented by solid lines, and non-accepting edges are dashed lines.

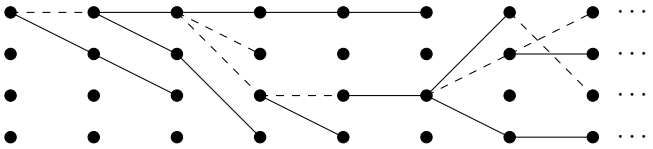


Figure 2 A tree-shaped Q-dag.

It is easy to see that a transducer can be used to reduce the question of acceptance from one deterministic automaton to another, as stated by the following lemma.

▶ **Lemma 22.** *For every deterministic Rabin automaton $\mathcal{A}$ with the input alphabet Γ, and every transducer $\mathcal{T}\colon \Sigma^{\mathbb{N}} \to \Gamma^{\mathbb{N}}$, there exists a deterministic Rabin automaton $\mathcal{A} \circ \mathcal{T}$ which accepts an infinite word $\alpha \in \Sigma^{\mathbb{N}}$ if and only if $\mathcal{A}$ accepts $\mathcal{T}(\alpha)$.*

One of the steps in the proof of Theorem 20, expressed by the lemma below, allows us to work with a fixed alphabet that depends only on the set of states of the given automaton $\mathcal{B}$. For that, we introduce a notion of a Q-dag. A Q-dag is an infinite word over the alphabet of transition matrices $[Q]$ of $\mathcal{B}$ that represents all the possible runs of $\mathcal{B}$ over a given infinite word, see Figure 1 (a formal definition will be given in the full paper).

▶ **Lemma 23.** *There exists a transducer $\mathcal{T}_1$ that inputs an infinite word $\alpha \in \Sigma^{\mathbb{N}}$ and outputs a Q-dag $\mathcal{T}_1(\alpha)$ such that $\mathcal{B}$ accepts α if and only if $\mathcal{T}_1(\alpha)$ contains an accepting path.*

This lemma is trivial—the transducer $\mathcal{T}_1$, after reading a finite word $w \in \Sigma^*$, stores in its state the set of states of $\mathcal{B}$ reachable from $q_{\mathrm{I}}^{\mathcal{B}}$ over w. The initial state of $\mathcal{T}_1$ is $\{q_{\mathrm{I}}\}$. Given a state $R \subseteq Q$ of $\mathcal{T}_1$ and a letter a, the transducer moves to the state

$$R' = \{q' \mid (q, a, q') \in \delta^{\mathcal{B}}\}$$

and outputs a letter $M \in [Q]$ such that $M(q, q') = M_a(q, q')$ if $q \in R$ and $M(q, q') = 0$ if $q \notin R$ (see Section 3 for the definition of M_a and $[Q]$). Clearly there is a computable bijection between the accepting runs of $\mathcal{B}$ over α and accepting paths in the Q-dag $\mathcal{T}_1(\alpha)$.

The next lemma shows that one can use a transducer to reduce general Q-dags to so-called *tree-shaped* Q-dags—the graph structure of such a word has the shape of a tree, see Figure 2.

▶ **Lemma 24.** *There exists a transducer $\mathcal{T}_2$ that inputs a Q-dag α' and outputs a tree-shaped Q-dag $\mathcal{T}_2(\alpha')$ such that α' contains an accepting path if and only if $\mathcal{T}_2(\alpha')$ contains an accepting path.*

To prove this lemma one uses a lexicographic order on paths in a given Q-dag. A crucial ingredient here is Bounded-width König's Lemma from Section 8. Additionally, we need to make sure that the graph to which Bounded-width König's Lemma is applied can be obtained using Δ^0_1-comprehension. For this purpose we use Σ^0_2-IND once again.

The proof of Theorem 20 is concluded by the following lemma and an application of Lemma 22.

▶ **Lemma 25.** *There exists a deterministic Rabin automaton $\mathcal{A}$ over the alphabet $[Q]$ that for every tree-shaped Q-dag $\alpha'' \in [Q]^{\mathbb{N}}$ accepts it if and only if α'' contains an accepting path.*

10 Conclusions and further work

In this work we have characterised the logical strength of Büchi's decidability theorem and related results over the theory RCA_0. We proved over RCA_0 that complementation for Büchi automata is equivalent to Σ^0_2-IND, as is the decidability of $\mathrm{MSO}(\mathbb{N}, \leq)$ (to the extent that this can be expressed).

Without Σ^0_2-IND, many aspects of automata on infinite words seem to make little sense (note, for instance, that the very concept of "a state occurs only finitely often" is Σ^0_2). The picture suggested by our work is that this minimal reasonability condition already suffices to prove all the basic results. This situation is completely different for automata on infinite trees, where the concepts also make sense already in $\mathsf{RCA}_0 + \Sigma^0_2$-$\mathsf{IND}$, but proving the complementation theorem or decidability of MSO requires much more [9].

We are thus led to the general question whether the entire theory of automata on infinite words requires exactly $\mathsf{RCA}_0 + \Sigma^0_2$-$\mathsf{IND}$. This includes in particular the following issues:

- Does McNaughton's determinisation theorem imply Σ^0_2-IND over RCA_0?
- How much axiomatic strength is needed to develop the algebraic approach to MSO ([14, Chapter II]), for instance to prove that Büchi-recognisability is equivalent to recognisability by finite Wilke algebras?
- What about developing the Wagner hierarchy (see [14, Chapter V.6])?
- Does $\mathsf{RCA}_0 + \Sigma^0_2$-$\mathsf{IND}$ prove the uniformisation theorem for automata, in the form: for a given automaton $\mathcal{A}$ over the alphabet $\{0,1\}^2$ such that $\forall X\, \exists Y\, (\mathcal{A}$ accepts $X \otimes Y)$, there exists an automaton $\mathcal{B}$ such that $\forall X\, \exists! Y$ (both $\mathcal{A}$ and $\mathcal{B}$ accept $X \otimes Y$) (see [15, Theorem 27])?

References

1 Christoph Schulte Althoff, Wolfgang Thomas, and Nico Wallmeier. Observations on determinization of Büchi automata. *Theor. Comput. Sci.*, 363(2):224–233, 2006. doi:10.1016/j.tcs.2006.07.026.

2 Mikołaj Bojańczyk. Lecture notes on languages, automata and computations, University of Warsaw, 2015. URL: http://www.mimuw.edu.pl/~bojan/20152016-2/jezyki-automaty-i-obliczenia-2/mcnaughtons-theorem.

3 J. Richard Büchi. On a decision method in restricted second order arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Logic, Methodology and Philosophy of Science. Proceeding of the 1960 International Congress*, pages 1–11. Stanford University Press, 1962.

4 Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1993. doi:10.1007/978-3-662-22156-3.

5 Denis R. Hirschfeldt. *Slicing the truth*, volume 28 of *Lecture Notes Series. Institute for Mathematical Sciences. National University of Singapore*. World Scientific, 2015.

6 Jeffry L. Hirst. *Combinatorics in Subsystems of Second Order Arithmetic*. PhD thesis, Pennsylvania State University, 1987.

7 Carl G. Jockusch, Jr. Ramsey's theorem and recursion theory. *J. Symbolic Logic*, 37:268–280, 1972.

8 Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *ICALP*, pages 724–735, 2008. doi:10.1007/978-3-540-70575-8_59.

9 Leszek Aleksander Kołodziejczyk and Henryk Michalewski. How unprovable is Rabin's decidability theorem? (accepted@LICS 2016). *CoRR*, abs/1508.06780, 2015. URL: `http://arxiv.org/abs/1508.06780`.

10 G. Kreisel. A variant to Hilbert's theory of the foundations of arithmetic. *British J. Philos. Sci.*, 4:107–129, 1953.

11 Jiayi Liu. RT^2_2 does not imply WKL_0. *J. Symbolic Logic*, 77(2):609–620, 2012. `doi:10.2178/jsl/1333566640`.

12 Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966. `doi:10.1016/S0019-9958(66)80013-X`.

13 David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141(1–2):69–107, 1995. `doi:10.1016/0304-3975(94)00214-4`.

14 Dominique Perrin and Jean-Éric Pin. *Infinite words : automata, semigroups, logic and games*. Pure and Applied Mathematics. Elsevier, London, San Diego (Calif.), 2004.

15 Alexander Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Information and Computation*, 205(6):870–889, 2007.

16 Saharon Shelah. The monadic theory of order. *Ann. of Math. (2)*, 102(3):379–419, 1975.

17 Stephen G. Simpson. *Subsystems of second order arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1999. `doi:10.1007/978-3-642-59971-2`.

The Height of Piecewise-Testable Languages with Applications in Logical Complexity*

Prateek Karandikar[1] and Philippe Schnoebelen[2]

1 **IRIF, Université Paris Diderot & LIPN, Université Paris 13, France**
2 **LSV, CNRS & ENS Cachan, Université Paris-Saclay, France**

Abstract

The height of a piecewise-testable language L is the maximum length of the words needed to define L by excluding and requiring given subwords. The height of L is an important descriptive complexity measure that has not yet been investigated in a systematic way. This paper develops a series of new techniques for bounding the height of finite languages and of languages obtained by taking closures by subwords, superwords and related operations.

As an application of these results, we show that $\mathsf{FO}^2(A^*, \sqsubseteq)$, the two-variable fragment of the first-order logic of sequences with the subword ordering, can only express piecewise-testable properties and has elementary complexity.

1998 ACM Subject Classification F.4.1 Mathematical Logic, F.4.3 Formal Languages, F.3.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases descriptive complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.37

1 Introduction

For two words u and v and some $n \in \mathbb{N}$, we write $u \sim_n v$ when u and v have the same (scattered) subwords[1] of length at most n. A language $L \subseteq A^*$ is *n-piecewise-testable* (or just "n-PT") if it is closed under $\sim_n$, or, equivalently, if it can be obtained as a boolean combination of *principal filters* of the form $A^*a_1A^*a_2A^* \cdots a_\ell A^*$ with $\ell \leqslant n$, and where $a_1, \ldots, a_\ell$ are letters from A. For example, with $A = \{a, b, c\}$, the language a^+b^* is 2-PT since it can be obtained as $A^*aA^* \cap \neg A^*cA^* \cap \neg A^*bA^*aA^*$. Thus a^+b^* can be described as "all words that have a but neither c nor ba as a subword". Finally, we say that L is piecewise-testable if it is n-piecewise-testable for some n and the smallest such n is called the piecewise-testability *height* of L, denoted $h(L)$ in this paper. We write PT for the class of piecewise-testable languages (over some alphabet A) and PT_n for the languages with height at most n, so that $\mathsf{PT}_0 \subseteq \mathsf{PT}_1 \subseteq \cdots \mathsf{PT}_n \subseteq \cdots \mathsf{PT}$ form a cumulative hierarchy of varieties of regular languages.

Piecewise-testable (PT) languages were introduced more than forty years ago in Simon's doctoral thesis (see [29, 27]) and have played an important role in the algebraic and logical theory of first-order definable languages, see [25, 4, 15] and the references therein. They also constitute an important class of simple regular languages with applications in learning theory [17], databases [2], linguistics [26], etc. The concept of PT languages has been extended to encompass trees [2], infinite words [24], pictures [23], etc.

* This work was partially supported by ANR grant ANR-14-CE28-0002 PACS.
[1] Or "subsequences", not to be confused with "factors".

© Prateek Karandikar and Philippe Schnoebelen;
licensed under Creative Commons License CC-BY

The height of piecewise-testable languages is a natural measure of descriptive complexity. Indeed, $h(L)$ coincides with the number of variables needed in a $\mathcal{B}\Sigma_1$ formula that defines L [4]. In this paper, the main question we address is "*how can one bound the height of PT languages obtained by natural language-theoretic operations?*" Since the height of these languages is a more robust measure than, say, their state complexity, it can be used advantageously in the complexity analysis of problems where PT languages are prominent. As a matter of fact, our results apply to open problems in the logic of subwords, see section 7.

Related work. The height of PT languages has been used to measure the difference between separable languages, see e.g. [9]. However the literature is quite poor on the question of estimating the height of PT languages. Algorithms that decide whether a regular language L (given e.g. by its canonical DFA $\mathcal{A}_L$) is PT usually provide a bound on $h(L)$ in terms of $\mathcal{A}_L$: recently Klíma and Polák showed that $h(L)$ is bounded by the depth of $\mathcal{A}_L$ [16]. The currently best bounds on $h(L)$ based on automata for L have been obtained by Masopust and Thomazo [22, 21].

When L is obtained by operations on other languages, very little is known about PT heights. It is clear that $h(A^* \setminus L) = h(L)$ and that $h(L \cup L') \leqslant \max(h(L), h(L'))$ but beyond boolean operations, quotients, and inverse morphisms, there are very few known ways of obtaining PT languages (see Appendix A).

Our contribution. We provide upper bounds on the PT height of finite languages and on PT-languages obtained by downward-closure (collecting all subwords of all words from some L), upward-closure, and some related operations (collecting words in L that are minimal wrt the subword ordering, etc.) We also show that the incomparability relation preserves piecewise-testability and bound the PT heights of the resulting languages. Crucially, we show that these bounds are *polynomial* when expressed in terms of the PT height of the arguments. One important tool is a *small-subword theorem* that shows how any long word u contains a short subword u' that is $\sim_n$-equivalent. Reasoning about subwords involves ad hoc techniques and leveraging the small-subword theorem to analyse downward-closures or incomparability languages turns out to be non-trivial. Subsequently, all the above results are used to prove that $\mathsf{FO}^2(A^*, \sqsubseteq)$, the two-variable logic of subwords, has elementary complexity. For this logic, the decidability proof in [14] did not come with an elementary complexity upper bound because the usual measures of complexity for regular languages can grow exponentially with each Boolean combination of upward and downward closures, and this is what prompted our investigation of PT heights.

Outline of the paper. Section 2 recalls the basic notions (subwords, $\sim_n$, ...) and gives some first bounds relating PT heights and canonical automata. Section 3 focuses on finite languages and develops our main tool: the small-subword theorem. Sections 4 and 5 give bounds for the height of PT languages obtained by upward and downward closures, while Section 6 considers the incomparability relation and the resulting PT heights. Finally, in Section 7 we apply these results to the complexity of $\mathsf{FO}^2(A^*, \sqsubseteq)$. Several proofs have been relegated to the Appendix, usually when the underlying techniques are not used in later developments.

2 Basic notions

We consider finite words $u, v, \ldots$ over a given finite alphabet A of letters like $a, b, \ldots$. Concatenation of words is written multiplicatively, with the empty word ε as unit. We freely use

regular expressions like $(ab)^* + (ba)^*$ to denote regular languages.

The length of a word u is written $|u|$ while, for a letter $a \in A$, $|u|_a$ denotes the number of occurrences of a in u. The set of all words over A is written A^* and for $\ell \in \mathbb{N}$ we use $A^{=\ell}$ and $A^{\leqslant \ell}$ to denote the subsets of all words of length ℓ and of length at most ℓ respectively.

A word v is a *factor* of u if there exist words u_1 and u_2 such that $u = u_1 v u_2$. If furthermore $u_1 = \varepsilon$ then v is a *prefix* of u and we write $v^{-1}u$ to denote the residual u_2. If $u_2 = \varepsilon$ then v is a *suffix* and $u\,v^{-1}$ is the residual.

Subwords. We say that a word u is a *subword* (i.e., a subsequence) of v, written $u \sqsubseteq v$, when u is some $a_1 \cdots a_n$ and v can be written as $v_0 a_1 v_1 \cdots a_n v_n$ for some $v_0, v_1, \ldots, v_n \in A^*$, e.g., $\varepsilon \sqsubseteq bba \sqsubseteq ababa$. We write $u \sqsubset v$ for the associated strict ordering, where $u \neq v$. Two words u and v are *incomparable* (with respect to the subword relation), denoted $u \perp v$, if $u \not\sqsubseteq v$ and $v \not\sqsubseteq u$. Factors are a special case of subwords.

With any $u \in A^*$ we associate the upward and downward closures, $\uparrow u$ and $\downarrow u$, given by[2]

$$\uparrow u \stackrel{\text{def}}{=} \{v \in A^* \mid u \sqsubseteq v\}\,, \qquad \downarrow u \stackrel{\text{def}}{=} \{v \in A^* \mid v \sqsubseteq u\}\,.$$

For example, $\downarrow ab = \{ab, a, b, \varepsilon\}$ and $\uparrow ab = A^*aA^*bA^*$. We also consider the strict superwords and subwords, with $\uparrow_< u \stackrel{\text{def}}{=} \{v \mid u \sqsubset v\}$ and $\downarrow_< u \stackrel{\text{def}}{=} \{v \mid v \sqsubset u\}$. This is generalised to the closures of whole languages, via e.g. $\uparrow L = \bigcup_{u \in L} \uparrow u$ and $\downarrow_< L = \bigcup_{u \in L} \downarrow_< u$. We say that a language L is *upward-closed* if $L = \uparrow L$, and *downward-closed* if $L = \downarrow L$. Note that a language is upward-closed if, and only if, its complement is downward-closed. It is known that upward-closed and downward-closed languages are regular (Haines Theorem [6], also a corollary of Higman's Lemma [8]) so $\uparrow L$, $\downarrow L$, $\uparrow_< L$ and $\downarrow_< L$ are regular for any L. Finally we further define

$$I(L) \stackrel{\text{def}}{=} \{u \in A^* \mid \exists v \in L : u \perp v\}\,.$$

Thus $I(L)$ collects all words that are incomparable with *some word* in L.

Simon's congruence and piecewise-testable languages. For $n \in \mathbb{N}$ and $u, v \in A^*$, we let

$$u \sim_n v \iff \downarrow u \cap A^{\leqslant n} = \downarrow v \cap A^{\leqslant n}\,. \qquad u \lesssim_n v \iff u \sim_n v \wedge u \sqsubseteq v\,. \tag{1}$$

Note that $\lesssim_n$ is stronger than $\sim_n$. Both relations are (pre)congruences: $u \sim_n v$ and $u' \sim_n v'$ imply $uu' \sim_n vv'$, while $u \lesssim_n v$ and $u' \lesssim_n v'$ imply $uu' \lesssim_n vv'$. The equivalence $\sim_n$ is called Simon's congruence of order n. We write $[u]_n$ for the equivalence class of $u \in A^*$ under $\sim_n$. Note that each $\sim_n$, for $n = 1, 2, \ldots$, has finite index.

There exist several characterisations of piecewise-testable languages: in the introduction we said that $L \subseteq A^*$ is n-piecewise-testable (or "n-PT") if it is a boolean combination of principal filters $A^*a_1A^*a_2A^* \cdots a_\ell A^*$ (i.e., of closures $\uparrow a_1 a_2 \cdots a_\ell$) with $\ell \leqslant n$. Equivalently, L is n-PT if it is a union $[u_1]_n \cup \cdots \cup [u_m]_n$ of $\sim_n$-classes. The first definition is convenient when we want to show that L is n-PT: we describe it in terms of required and excluded subwords and check the length of these subwords, as when we showed that a^+b^* is 2-PT. The second characterisation is convenient when we want to show that L is not n-PT: by exhibiting two words $u \sim_n v$ such that $u \in L$ and $v \notin L$, one proves that L is not saturated by $\sim_n$. E.g., a^+b^* is not 1-PT since $ab \sim_1 ba$ while only ab is in a^+b^*.

[2] The definition of $\uparrow u$ involves an implicit alphabet A that will always be clear from the context.

When we abstract away from n, we said that a language $L \subseteq A^*$ is piecewise-testable (or PT) if it is n-PT for some n. Other characterisations are: L is PT iff its syntactic monoid is $\mathcal{J}$-trivial (Simon's Theorem), iff it is definable in the $\mathcal{B}\Sigma_1$ fragment of the first-order logic over words [4], iff its canonical DFA is acyclic and locally confluent [16].

Note that if L is n-PT, it is also m-PT for any $m > n$. We write $h(L)$ for the smallest n – called the "height of L" – such that L is n-PT, letting $h(L) = \infty$ when L is not PT.

The following properties will be useful:

▶ **Lemma 2.1.** *For all $u, v \in A^*$ and $a \in A$:*
(1) *If $u \lesssim_n v$ then $u \sim_n w$ for all $w \in A^*$ such that $u \sqsubseteq w \sqsubseteq v$.*
(2) *If $u \sim_n v$ then there exists $w \in A^*$ such that $u \lesssim_n w$ and $v \lesssim_n w$.*
(3) *If $uv \sim_n uav$ then $uv \sim_n ua^\ell v$ for all $\ell \in \mathbb{N}$.*
(4) *Every equivalence class of $\sim_n$ is a singleton or is infinite.*

Proof. (1) is by combining Eq. (1) and $\downarrow u \subseteq \downarrow w \subseteq \downarrow v$; (2) is Lemma 6 from [29]; (3) is in the proof of [27, Coro. 2.8]; (4) follows from (1), (2) and (3). ◀

Constructing PT languages. Recall that PT languages constitute a subvariety of the dot-depth 1 languages, themselves a subvariety of the star-free languages, themselves a subvariety of the regular languages [4]. As such, all classes PT_n for $n \in \mathbb{N}$, as well as PT, are closed under union, intersection, complementation, inverse morphisms and quotients (left and right residuals). These properties lead to (in)equations like

$$h(L \cup L') \leqslant \max(h(L), h(L'))\,, \qquad h(\neg L) = h(L)\,, \tag{2}$$
$$h(u^{-1}L) \leqslant h(L)\,, \qquad h(Lv^{-1}) \leqslant h(L)\,, \tag{3}$$
$$h(\rho^{-1}(L)) \leqslant h(L) \text{ when } \rho : A^* \to B^* \text{ is a morphism,} \tag{4}$$

that can be used to bound the height of the PT languages we construct. See Appendix B for a proof of Eq. (4).

Relating PT height and state complexity. For regular languages, a standard measure of descriptive complexity is *state complexity*, denoted $sc(L)$, and defined as the number of states of the canonical DFA for L.

The bounds we just listed let us contrast the height of a PT language with its state complexity. For PT languages, $h(L)$ is smaller than or equal to $sc(L)$ (equality occurs e.g. when $L = \{a^\ell\}$) since $sc(L)$ bounds the depth of the automaton, i.e., the maximum length of a simple path from the initial to some final state, which in turns bounds $h(L)$ [16, 22].

In the other direction, we can prove:

▶ **Theorem 2.2.** *Let A be an alphabet of size k with $k > 1$. Suppose $L \subseteq A^*$ is n-PT. Then the canonical DFA for L has at most m states,*[3] *where*

$$\log m = k \left(\frac{n + 2k - 3}{k - 1} \right)^{k-1} \log n \log k\,.$$

Here log *means* log *to the base* 2. *Thus, for fixed k, $sc(L)$ is in $2^{O(n^{k-1} \log n)}$, where $n = h(L)$.*

[3] It is shown in [22] that the depth (not the size) of the canonical DFA is bounded by $\binom{n+k}{n} - 1$.

Proof. We build a DFA for L which remembers the equivalence class under $\sim_n$ of the word it has read so far. This is possible because for all $w \in A^*$ and $a \in A$, the class $[wa]_n$ of wa is determined by $[w]_n$ and a. The initial state is $[\varepsilon]_n$, and the final states are all the classes $[u]_n$ which are a subset of L. In [11] we showed that the number of equivalence classes of $\sim_n$ is bounded by m. ◀

The general situation is that $h(L)$ can be much smaller than $sc(L)$ as we see in the following sections. More importantly, $h(L)$ is more robust than $sc(L)$ and, for example, state complexity will usually increase (sometimes exponentially) when constructing a regular language with boolean combinations of simpler languages[4] while PT height will not increase.[5]

More constructions for PT languages. Two simple but important constructions that provide PT languages are the closures by subwords and superwords, $\uparrow L$ and $\downarrow L$, defined above. Every upward-closed language is PT since it is the union of finitely many languages of the form $\uparrow u$ (by Higman's Lemma [18]). Every downward-closed language is PT too since its complement is upward-closed. Analysing the height of these PT languages is the topic of Sections 4 and 5.

We are not aware of more piecewise-testability preserving operations on languages in the literature. Let us recall that PT languages are not closed under concatenation (even just $L \mapsto a.L$), Kleene star, shuffle product, conjugacy, and simple operations like renamings (length-preserving morphisms) or the erasing of one letter, see Appendix A for details.

In view of this, it was a good surprise to discover that $I(L)$ is PT when L is. Bounding its height requires a non-trivial ad hoc proof and is the topic of Section 6.

3 PT height of words and the small-subword theorem

Our starting point is an analysis of the PT height of single words. It is clear that any singleton language $\{u\}$ is PT since $\{u\} = \uparrow u \smallsetminus \bigcup_{v \in \{u\} \sqcup\!\sqcup A} \uparrow v$, which entails $h(\{u\}) \leqslant |u| + 1$. Here $\{u\} \sqcup\!\sqcup A$ is a *shuffle product*, collecting all the shuffles of u with a letter from A. In other words, $\{u\} \sqcup\!\sqcup A = \{v \ : \ u \sqsubseteq v \wedge |v| = |u| + 1\}$. Below we often omit set-theoretical parentheses when denoting singletons, writing e.g. "$h(u)$" or "$u \sqcup\!\sqcup A$".

The PT height of a singleton language can be computed in time $O((|u| + |A|) \cdot |u| \cdot |A|)$, see Appendix C. This can be used to compute the PT height of finite languages: for such languages, the inequality in Eq. (2) becomes

$$h(\{u_1, \ldots, u_m\}) = \max\{h(u_1), \ldots, h(u_m)\} \,. \tag{5}$$

Indeed, $h(\{u_1, \ldots, u_m\}) = n$ implies $[u_i]_n \subseteq \{u_1, \ldots, u_m\}$ for any i. Thus $[u_i]_n$ is a singleton in view of Lemma 2.1.4. Hence $[u_i]_n = \{u_i\}$ and $h(u_i) \leqslant n$.

The $|u| + 1$ upper bound for $h(u)$ is reached for $u = a^\ell$ (to see that $h(a^\ell) > \ell$, one notes that $\{a^\ell\}$ is not closed under $\sim_\ell$ since $a^\ell \sim_\ell a^{\ell+1}$). However, words on more than one letter can generally be described within some PT height lower than their length. For example

$$\{aabb\} = (\uparrow aa \cap \uparrow bb) \smallsetminus (\uparrow ba \cup \uparrow aaa \cup \uparrow bbb) \,,$$

[4] Such combinatorial explosions also occur when restricting to piecewise-testable languages [12].

[5] This robustness is not restricted to boolean operations: write $rev(u)$ for the *reversal* of u, e.g., $rev(abc) = cba$ and extend to languages. It is clear that $rev(L)$ is n-PT when L is – indeed $rev(\uparrow u) = \uparrow rev(u)$, $rev(L \cup L') = rev(L) \cup rev(L')$, and $rev(A^* \smallsetminus L) = A^* \smallsetminus rev(L)$ – but $sc(rev(L))$ cannot be bounded by a polynomial of $sc(L)$, even in the case of finite, hence PT, languages [28].

showing $h(aabb) \leqslant 3$. (Note that $h(aabb) > 2$ since $aabbb \sim_2 aabb$.) It turns out that the PT height of words can be much lower than their length as we now show.

Words with low PT height. We now introduce a family of words with "low PT height" that will be used repeatedly in later sections. Let $A_k = \{a_1, \ldots, a_k\}$ be a k-letter alphabet. We define a word $U_k \in A_k^*$ by induction on k and parameterized by a parameter $\eta \in \mathbb{N}$. We let $U_0 \stackrel{\text{def}}{=} \varepsilon$ and, for $k > 0$, $U_k \stackrel{\text{def}}{=} (U_{k-1}a_k)^\eta U_{k-1}$. For example, for $\eta = 3$ and $k = 2$, one has $U_2 = a_1a_1a_1a_2a_1a_1a_1a_2a_1a_1a_1a_2a_1a_1a_1$.

▶ **Proposition 3.1.** *For $k \geqslant 0$, $|U_k| = (\eta+1)^k - 1$ and $h(U_k) = k\eta + 1$.*

Proof sketch. An easy induction on k shows that $|U_k| = (\eta+1)^k - 1$. To show $h(U_k) = k\eta+1$ we use auxiliary languages $P_k, N_k \subseteq A_k^*$ defined inductively by:

$$P_0 = \{\varepsilon\}, \quad N_0 = \varnothing,$$

and, for $k > 0$,

$$P_k = \{a_k^i v a_k^{\eta-i} \mid 0 \leqslant i \leqslant \eta \wedge v \in P_{k-1}\},$$
$$N_k = \{a_k^{\eta+1}\} \cup \{a_k^i w a_k^{\eta-i} \mid 0 \leqslant i \leqslant \eta \wedge w \in N_{k-1}\}.$$

We now claim that for any $k \in \mathbb{N}$ and $u \in A_k^*$:

$$\Big(\bigwedge_{v\in P_k} v \sqsubseteq u\Big) \wedge \Big(\bigwedge_{w\in N_k} w \not\sqsubseteq u\Big) \iff u = U_k. \tag{6}$$

(See Appendix D for a proof.) Thus $h(U_k) \leqslant k\eta + 1$ since the words in P_k have length $k\eta$ and the words in N_k have length at most $k\eta + 1$.

It remains to show that $h(U_k) > k\eta$, i.e., that $\{U_k\}$ is not closed under $\sim_{k\eta}$: for this it is enough to note that $U_k \sim_{k\eta} U_k a_1$ using [29, Lemma 3]. ◀

For later use we also record the following bounds (see Appendix E):

$$h(\downarrow U_k) = \eta(\eta+1)^{k-1} + 1\,. \tag{7}$$

Rich words and rich factorizations. Assume a fixed k-letter alphabet A. We say that a word u is *rich* if all the k letters of A occur in it, and that it is *poor* otherwise. For $\ell \in \mathbb{N}$, we further say that u is ℓ-rich if it can be written as a concatenation $u = u_1 \cdots u_\ell u'$ where the ℓ factors $u_1, \ldots, u_\ell$ are rich.

The *richness* of u is the largest $\ell \in \mathbb{N}$ such that u is ℓ-rich. Note that having $|u|_a \geqslant \ell$ for all letters $a \in A$ does not imply that u is ℓ-rich.

▶ **Lemma 3.2** (See Appendix F). *If u_1 and u_2 are respectively ℓ_1-rich and ℓ_2-rich, then $v \sim_n v'$ implies $u_1vu_2 \sim_{\ell_1+n+\ell_2} u_1v'u_2$.*

The *rich factorization* of $u \in A^*$ is the decomposition $u = u_1a_1 \cdots u_ma_mv$ defined by induction in the following way: if u is poor, we let $m = 0$ and $v = u$; otherwise u is rich, we let u_1a_1 (with $a_1 \in A$) be the shortest prefix of u that is rich and let $u_2a_2 \cdots u_ma_mv$ be the rich factorization of the remaining suffix $(u_1a_1)^{-1}u$. By construction m is the richness of u. E.g., assuming $k = 3$ and $A = \{a, b, c\}$, the following is a rich factorization with $m = 2$:

$$\overbrace{bbaaabbccccaabbbaa}^{u} = \overbrace{bbaaabb}^{u_1} \cdot c \cdot \overbrace{cccaa}^{u_2} \cdot b \cdot \overbrace{bbaa}^{v}$$

Note that, by construction, $u_1, \ldots, u_m$ and v are poor.

▶ **Lemma 3.3** (See Appendix F). *Consider two words u, u' of richness m and with rich factorizations $u = u_1 a_1 \cdots u_m a_m v$ and $u' = u'_1 a_1 \cdots u'_m a_m v'$. Suppose that $v \sim_n v'$ and that $u_i \sim_{n+1} u'_i$ for all $i = 1, \ldots, m$. Then $u \sim_{n+m} u'$.*

The small-subword theorem. Our next result is used to prove lower bounds on the PT height of long words. It will be used repeatedly in the course of this paper.

For $k = 1, 2, \ldots$ define $f_k : \mathbb{N} \to \mathbb{N}$ by induction on k with

$$f_1(n) = n\,, \tag{8}$$

$$f_{k+1}(n) = \max_{0 \leqslant m \leqslant n} m f_k(n+1-m) + m + f_k(n-m)\,. \tag{9}$$

The definition of f_k is only used in the proof of Theorem 3.4. In the rest of the paper, we simplify things by relying on the following upper bound (proved in [13, Prop. 4.4]):

$$f_k(n) \leqslant \Big(\frac{n+2k-1}{k}\Big)^k - 1 < \Big(\frac{n}{k}+2\Big)^k\,. \tag{10}$$

▶ **Theorem 3.4** (Small-subword Theorem). *Let $k = |A|$. For all $u \in A^*$ and $n \in \mathbb{N}$ there exists some $v \in A^*$ with $v \lesssim_n u$ and such that $|v| \leqslant f_k(n)$.*

Proof. By induction on k, the size of the alphabet.

With the base case, $k = 1$, we consider a unary alphabet $A = \{a\}$ and u is $a^{|u|}$. Now $a^\ell \sim_n u$ iff $\ell = |u| < n$ or $\ell \geqslant n \leqslant |u|$. So taking $v = a^\ell$ for $\ell = \min(n, |u|)$ proves the claim.

When $k > 1$ we consider the rich factorization $u = u_1 a_1 u_2 a_2 \cdots u_m a_m u'$ of u. Let $n' = \max(n+1-m, 1)$. Every u_i is a word on the subalphabet $A \setminus \{a_i\}$. Hence by induction hypothesis there exists $v_i \sqsubseteq u_i$ with $|v_i| \leqslant f_{k-1}(n')$ and $v_i \sim_{n'} u_i$, entailing $u_i a_i \sim_{n'} v_i a_i$. Similarly, the induction hypothesis entails the existence of some $v' \sqsubseteq u'$ with $v' \sim_{n'-1} u'$ and $|v'| \leqslant f_{k-1}(n'-1)$. Note that in these inductive steps we use a length bound obtained with f_{k-1} by using the fact that $u_1, \ldots, u_m$ and u', being poor, use at most $k-1$ letters from A.

We now consider two cases. If $m \leqslant n-1$, we let $v = v_1 a_1 \cdots v_m a_m v'$, so that $v \sqsubseteq u$ and $|v| \leqslant m f_{k-1}(n') + m + f_{k-1}(n'-1)$. We deduce $|v| \leqslant f_k(n)$ using Eq. (9) and since $n' = n+1-m$. That $v \sim_n u$, hence $v \lesssim_n u$, is an application of Lemma 3.3: $v_1 a_1 \cdots v_m a_m v'$ is indeed the rich decomposition of v since $n' \geqslant 2$, $v' \sim_{n'-1} u'$, and $v_i \sim_{n'} u_i$ for $i = 1, \ldots, m$.

If $m \geqslant n$, then u is n-rich and its subwords include all words of length at most n. It is easy to extract some n-rich subword v of u that only uses kn letters. Now $v \sim_n u$ since both u and v are n-rich, entailing $v \lesssim_n u$. One also checks that $|v| = kn \leqslant f_k(n)$. ◀

Note that the bound $f_k(n)$ in Theorem 3.4 does not depend on u.

We can already apply the small-subword theorem to the case of finite languages.

▶ **Proposition 3.5** (Finite languages). *Suppose $L \subseteq A^*$ is finite and nonempty with $|A| = k$. Let ℓ be the length of the longest word in L. Then $1 + k(\ell^{1/k} - 2) < h(L) \leqslant \ell + 1$.*

Proof. Thanks to Eq. (5), it is enough to consider the case where $L = \{u\}$ is a singleton. So assume $h(L) = h(u) = n$ and $|u| = \ell$. The small-subword theorem says that $u \sim_n v$ for some short v but necessarily $v = u$ since $[u]_n$ is a singleton, hence $\ell \leqslant f_k(n)$. Using Eq. (10) one gets $\ell \leqslant f_k(n) < \big(\frac{n+2k-1}{k}\big)^k$. This gives $n > 1 + k(\ell^{1/k} - 2)$. The upper bound $h(L) \leqslant \ell + 1$ was observed earlier. ◀

We already noted that the upper bound is tight. The lower bound is quite good: for U_k seen above, $\ell = (\eta+1)^k - 1$, so that $\ell \leqslant \big(\frac{n+2k-1}{k}\big)^k - 1$ gives $n = h(U_k) \geqslant k\eta - k + 1$ while we know $h(U_k) = k\eta + 1$.

4 Upward closures

It is known that $\uparrow L$ is PT for any L. Related languages are $\uparrow_< L$ (used in Section 7) and $\min(L) \stackrel{\text{def}}{=} \{u \in L \mid \forall v \in L : v \not\sqsubset u\}$. This section provides bounds on the PT height of languages obtained by such constructions.

We first note that, in the special case of singletons, the PT height of $\uparrow L$ and $I(L)$ always coincide with word length:[6]

▶ **Proposition 4.1.** *For any $u \in A^*$*

$$h(\uparrow u) = |u|\,, \qquad h(I(u)) = h(\uparrow u \cup \downarrow u) = \begin{cases} |u| & \text{if } |A| \geqslant 2, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Proof. Let $\ell = |u|$. Obviously $h(\uparrow u) \leqslant \ell$ and the point is to prove $h(\uparrow u) > \ell - 1$. For this we assume $\ell > 0$ and write $u = a_1 \cdots a_\ell$. With a letter $a \in A$ we associate a word π_a of length $|A|$ that lists all the letters of A exactly once *and ends with* a. E.g. $\pi_b = acdb$ works when $A = \{a, b, c, d\}$. Let now $v = \pi_{a_1}\pi_{a_2} \cdots \pi_{a_{\ell-1}}$ and $v' = v \cdot a_\ell$. Then $v \sim_{\ell-1} v'$ since v has all subwords of length $\ell - 1$. However $u \not\sqsubset v$ and $u \sqsubseteq v'$ hence $\uparrow u$ is not closed under $\sim_{\ell-1}$.

Now for $I(u)$: we note that *in the case of singletons* we can write $I(u) = A^* \setminus (\uparrow u \cup \downarrow_< u)$, from which $h(I(u)) \leqslant \ell$ follows since all the finitely many words in $\downarrow_< u$ have length at most $\ell - 1$. To show $h(I(u)) > \ell - 1$ when $|A| \geqslant 2$, we assume $\ell > 1$ and use v and v' again: $v' \notin I(u)$ while $v \in I(u)$ hence $I(u)$ is not closed under $\sim_{\ell-1}$. Finally, when $|A| < 2$ or $\ell = 0$, $I(u) = \varnothing$, while when $\ell = 1$ and $|A| \geqslant 2$, $I(u)$ is neither $\varnothing$ nor A^*. ◀

▶ **Corollary 4.2.** *For any $L \subseteq A^*$ and $m \in \mathbb{N}$, if all words in $\min(L)$ have length bounded by m, then $\uparrow L$ is m-PT while $\uparrow_< L$ and $\min(L)$ are $(m+1)$-PT.*

Proof. Recall that $\min(L)$ is finite by Higman's Lemma. Then note that $\uparrow L = \uparrow \min(L)$ and that $\uparrow_< L = (\uparrow L) \setminus \min(L)$. Use $h(u) \leqslant |u| + 1$ from Section 3. ◀

This can be immediately applied to languages given by automata or grammars.

▶ **Theorem 4.3** (Upward closures of regular and context-free languages)**.**

(1) *If L is accepted by a nondeterministic automaton (a NFA) having depth m, then $\uparrow L$ is m-PT while $\uparrow_< L$ and $\min(L)$ are $(m+1)$-PT.*

(2) *The same holds if L is accepted by a context-free grammar (a CFG) when we let $m = \ell^N$ where N is the number of nonterminal symbols and ℓ is the maximum length for the right-hand side of production rules.*

Proof sketch. (1) A word accepted by the NFA is minimal wrt $\sqsubseteq$ only if it is accepted along an acyclic path. (2) A word generated by the CFG is minimal wrt $\sqsubseteq$ only if any nonterminal appears at most once along any branch of its derivation tree. ◀

The bounds in Theorem 4.3 can be reached, e.g., for L a singleton of the form $\{a^m\}$.

For our applications, we are interested in expressing $h(\uparrow L)$ in terms of $h(L)$, assuming that L is PT.

▶ **Theorem 4.4** (Upward closures of PT languages)**.** *Suppose $L \subseteq A^*$ is n-PT and $|A| = k$. Let $m = f_k(n)$. Then $\uparrow L$ is m-PT, while $\uparrow_< L$ and $\min(L)$ are $(m+1)$-PT.*

[6] This phenomenon does not extend to the other operations nor to finite sets.

Proof. By the small-subword theorem, and since L is closed under $\sim_n$, the minimal elements of L have length bounded by m. Then Corollary 4.2 applies. ◀

▶ Remark 4.5. The upper bound in Theorem 4.4 is quite good: for any $k, \eta \geqslant 1$, the language $L = \{U_k\}$ has $h(U_k) = n = k\eta + 1$ so that Theorem 4.4 with Eq. (10) give $h(\uparrow U_k) \leqslant f_k(n) < (\eta + 2)^k$. On the other hand we know that $h(\uparrow U_k) = (\eta + 1)^k - 1$ by Proposition 4.1.

5 Downward closures

We now move to downward closures. It is known that, for any $L \subseteq A^*$, $\downarrow L$ and $\downarrow_< L$ are PT since they are the complement of upward-closed languages. Our strategy for bounding $h(\downarrow L)$ is to approximate L by finitely many D-products.

A *D-product* is a regular expression P of the form $E_1 \cdot E_2 \cdots E_\ell$ where every E_i is either of the form B^* for a subalphabet $B \subseteq A$ (B^* is called a *star factor* of P), or a single letter $a \in A$ (a *letter factor*). We say that ℓ is the length of P.

▶ **Proposition 5.1.** *If P is a D-product of length ℓ, $h(\downarrow P) \leqslant \ell + 1$ and $h(\downarrow_< P) \leqslant \ell + 1$.*

Proof. Let P' be the regular expression obtained from P by replacing any letter factor a by $(a + \varepsilon)$ so that $P' = \downarrow P$. Now any residual $w^{-1}P'$ of P' is either the empty language $\varnothing$, or corresponds to a suffix P'' of P'. This is shown by induction on the length of suffixes, using

$$b^{-1}\big[(a+\varepsilon)P''\big] = \begin{cases} P'' & \text{if } b = a, \\ b^{-1}P'' & \text{otherwise,} \end{cases} \qquad b^{-1}\big[B^*P''\big] = \begin{cases} B^*P'' & \text{if } b \in B, \\ b^{-1}P'' & \text{otherwise,} \end{cases}$$

and $a^{-1}\varepsilon = \varnothing$ for the last suffix, i.e., the empty product. (Note that the correctness of the first equality when $b = a$, and of the second equality when $b \in B$, rely on $b^{-1}P'' \subseteq P''$: this holds because P'' is downward-closed.) Thus P' has at most $\ell + 1$ distinct non-empty residuals, i.e., the canonical DFA for P' has at most $\ell + 1$ productive states, hence has depth at most $\ell + 1$. We now apply Theorems 1 and 2 from [16] and conclude that $h(\downarrow P) \leqslant \ell + 1$.

For $\downarrow_< P$ very little need to be changed. If P contains at least one star factor then $\downarrow_< P$ and $\downarrow P$ coincide. If P only contains letter factors then P denotes a singleton $\{u\}$ with $|u| = \ell$. Then $\downarrow_< P$ is a finite set of words of length at most $\ell - 1$, entailing $h(\downarrow_< P) \leqslant \ell$. ◀

The bounds in Proposition 5.1 can be reached, e.g., for $P = a \cdots a$.

▶ **Corollary 5.2.** *If $L \subseteq \bigcup_i P_i \subseteq \downarrow L$ for a family $(P_i)_i$ of D-products of length at most ℓ, then $h(\downarrow L) \leqslant \ell + 1$ and $h(\downarrow_< L) \leqslant \ell + 1$.*

Proof. Obviously $\downarrow L = \bigcup_i \downarrow P_i$ and the union is finite since there are only finitely many D-products of bounded length. ◀

This can be immediately applied to languages given by automata or grammars.

▶ **Theorem 5.3** (Downward-closures of regular and context-free languages)**.**

(1) *If L is accepted by a nondeterministic automaton (a NFA) having depth m, then $\downarrow L$ and $\downarrow_< L$ are ℓ-PT for $\ell = 2m + 2$.*

(2) *The same holds if L is accepted by a CFG in quadratic normal form (a QNF, see [1]) with N nonterminals and $\ell = 4 \cdot 3^{N-1} + 2$.*

Proof sketch. (1) For a word $u \in L$ we consider the cycles in an accepting path on u. This leads to a factoring $u = u_0 a_1 u_1 a_2 \cdots a_p u_p$ of u such that the accepting path is some $q_0 \xrightarrow{u_0} q_0 \xrightarrow{a_1} q_1 \xrightarrow{u_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{p-1} \xrightarrow{a_p} q_p \xrightarrow{u_p} q_p$ with $q_0, q_1, \ldots, q_p$ all different. Then $p \leqslant m$. Let now $B_i \subseteq A$ be the set of letters occurring in u_i and define $P_u \stackrel{\text{def}}{=} B_0^* a_1 B_1^* a_2 \ldots B_{p-1}^* a_p B_p^*$. Then $u \in P_u$ and $P_u \subseteq {\downarrow}L$. Finally, $L \subseteq \bigcup_{u \in L} P_u \subseteq {\downarrow}L$ and each P_u has length $\leqslant 2m+1$. (2) Bachmeier et al. showed that there is an NFA for ${\downarrow}L$ having $2 \cdot 3^{N-1}$ states [1]. ◀

For our applications, we are interested in bounding $h({\downarrow}L)$ in terms of $h(L)$ when L is PT.

▶ **Theorem 5.4** (Downward closures of PT languages)**.** *Suppose $L \subseteq A^*$ is n-PT and $|A| = k$. Let $m = f_k(n)$. Then ${\downarrow}L$ and ${\downarrow_<}L$ are $(k+1)(m+1)$-PT.*

The rest of this section is devoted to the proof of Theorem 5.4. Our strategy is to approximate L by D-products, this time relying on the fact that L is closed under $\sim_n$.

Recall Lemma 2.1.3 stating that, given two words u, v and a letter $a \in A$, $uv \sim_n uav$ entails $uv \sim_n ua^\ell v$ for all $\ell \in \mathbb{N}$. We express this as "$uav \in [uv]_n$ implies $ua^*v \subseteq [uv]_n$" and call it a *pumping property* of PT classes. We now establish more general pumping properties.

▶ **Lemma 5.5.** *If $uB^*C^*B^*v \subseteq [uv]_n$, where $B, C \subseteq A$ are subalphabets, then $u(B \cup C)^* v \subseteq [uv]_n$.*

Proof idea. We prove that for any $m \in \mathbb{N}$, for any $w \in B^*(C^*B^*)^m$, for any $s \in A^{\leqslant n}$, $s \sqsubseteq uwv$ implies $s \sqsubseteq uv$. The proof is by induction on m, knowing that the claim holds by assumption for $m \leqslant 1$. See Appendix G for details. ◀

Going on we can show that $uab_1b_2\cdots b_m av \sim_n uv$ entails $uwv \sim_n uv$ for all $w \in (a+b_1)^*(a+b_2)^* \cdots (a+b_m)^*$, hence the two surrounding a's can join any surrounded letter.

▶ **Lemma 5.6** (See Appendix G)**.** *Suppose $L_1 B_1^* B_2^* \cdots B_\ell^* L_2 \subseteq [u]_n$ for some languages $L_1, L_2 \subseteq A^*$ and subalphabets $B_1, B_2, \ldots, B_\ell \subseteq A$ with $\ell \geqslant 3$. If $a \in B_1 \cap B_\ell$ then, letting $B_i' = B_i \cup \{a\}$, $L_1 {B'}_1^* {B'}_2^* \cdots {B'}_\ell^* L_2 \subseteq [u]_n$.*

There remains to bound the products that cannot be simplified by the above Lemmas.

▶ **Lemma 5.7.** *Suppose A is a finite set and $E_1, E_2, \ldots, E_\ell$ are $\ell > 1$ subsets of A such that the following hold:*

- *for all $1 \leqslant i < \ell$, $E_i \not\subseteq E_{i+1}$ and $E_{i+1} \not\subseteq E_i$;*
- *for all $a \in A$ and $1 \leqslant i < j \leqslant \ell$, if $a \in E_i \cap E_j$, then $a \in E_k$ for all $i \leqslant k \leqslant j$.*

Then $\ell \leqslant |A|$.

Proof. Note that by the first condition, each E_i is nonempty. Define $E_0 = E_{\ell+1} = \varnothing$. For $0 \leqslant i \leqslant \ell$, define $F_i = E_i \triangle E_{i+1}$, where $\triangle$ denotes symmetric difference. Now F_0 and F_ℓ have size at least 1, and by the first condition, every other F_i has size at least 2. Thus $\sum_i |F_i| \geqslant 2\ell$. By the second condition, any $a \in A$ occurs in at most two F_i's, thus $\sum_i |F_i| \leqslant 2|A|$. So we conclude $2\ell \leqslant 2|A|$. ◀

▶ **Lemma 5.8.** *Let $L \subseteq A^*$ be n-PT. Let $k = |A|$ and $m = f_k(n)$. For every $u \in L$ there is a D-product P_u of length $\ell \leqslant mk + m + k$ such that $u \in P_u \subseteq L$.*

Proof idea. A formal proof is given in Appendix G and we just outline it here: Assume $u \in L$. By the small-subword theorem there exists a subword $v = a_1 \cdots a_\ell$ of u with $v \sim_n u$ and $\ell = |v| \leqslant m$. So u is v plus some added letters. By Lemma 2.1.3, all these added (occurrences of) letters can be pumped, yielding a D-product with $u \in P \subseteq [u]_n$. Applying Lemma 5.6 and further simplifications yields a shorter D-product with $P \subseteq P_u \subseteq [u]_n$. Since P_u cannot be further simplified, Lemma 5.7 can be used to bound its size. ◀

We may now conclude:

Proof of Theorem 5.4. With Lemma 5.8 we obtain $L = \bigcup_{u \in L} P_u$ where each P_u has length bounded by $km + k + m$. We can then apply Proposition 5.1. ◀

▶ Remark 5.9. The upper bound in Theorem 5.4 is quite good: for any $k, \eta \geqslant 1$, the language $L = \{U_k\}$ from Proposition 3.1 has $h(U_k) = n = k\eta + 1$ so that Theorem 5.4 gives $h(\downarrow U_k) < (k+1)(\eta+2)^k$. On the other hand we know that $h(\downarrow U_k) = \eta(\eta+1)^{k-1} + 1$ by Eq. (7).

6 Piecewise-testability and PT height for $I(L)$

Recall that $I(L)$ is the set of words which are incomparable (under $\sqsubseteq$) with *some word* in L. In this section we prove the following result.

▶ **Theorem 6.1.** *Suppose $L \subseteq A^*$ is n-PT and $|A| = k$. Let $m = f_k(n)$. Then $I(L)$ is $(m+1)$-PT.*

It is not too difficult to show the regularity of $I(L)$ when L is regular, and this can be done using standard automata-theoretical techniques. Indeed, it can be shown that the incomparability relation $\perp$ is a rational relation [14].

Showing that I also preserves piecewise-testability requires more work. For such questions, I does not behave as simply as the other pre-images we considered before. In particular, we observe that $I(L)$ is not necessarily PT when L is regular. For example, taking $A = \{a, b, c\}$ and letting

$$L = (abc)^*(\varepsilon + a + ab) = \{\varepsilon, a, ab, abc, abca, abcab, \ldots\}$$

gives a language that is totally ordered by $\sqsubseteq$ and contains one word of each length, so that $I(L) = A^* \smallsetminus L$, which is not PT since L is not.

Similarly, $I(L)$ is not necessarily regular when L is not. For example, taking $A = \{a, b\}$ and

$$L = \{a^\ell b^\ell(\varepsilon + b) \mid \ell \in \mathbb{N}\} = \{\varepsilon, b, ab, abb, aabb, a^2b^3, a^3b^3, \ldots\}\,.$$

Again L is totally ordered by $\sqsubseteq$ and contains one word of each length. Hence $I(L) = A^* \smallsetminus L$, which is not regular.

The above examples illustrate our strategy for proving Theorem 6.1: if a language L is totally ordered by $\sqsubseteq$ then $I(L) \cap L = \varnothing$, or equivalently $I(L) \subseteq A^* \smallsetminus L$. Similarly, if L contains at least two words having same length ℓ then $I(L)$ contains all words of length ℓ.

We now proceed with a more formal proof. For technical convenience we introduce a dual construct:

$$C(L) \stackrel{\text{def}}{=} \{u \in A^* \mid L \subseteq \uparrow u \cup \downarrow u\}\,.$$

Note that $C(L)$ coincides with $A^* \smallsetminus I(L)$ and that $C(L \cup L') = C(L) \cap C(L')$. We find it easier to analyse $C(L)$ instead of $I(L)$, but these two languages have the same PT height.

As we just hinted at, it is useful to think of the "layers" $L \cap A^{=\ell} = \{w \in L : |w| = \ell\}$ of L, and check whether they contain 0, 1 or more words (we say that the layer is *empty*, *singular*, or *populous*). Observe that if $L \cap A^{=\ell}$ is populous then $C(L) \cap A^{=\ell}$ is empty.

For the rest of this section, we consider a fixed $n \geqslant 1$ and let $m = f_k(n)$. We start with a technical lemma: write $u \lesssim^1_n v$ when $u \lesssim_n v$ and $|v| = |u| + 1$, i.e., v is u with one letter added in a way that is compatible with $\sim_n$. Note that $\lesssim_n$ is the transitive closure of $\lesssim^1_n$.

▶ **Lemma 6.2.** *Let $u, v, w \in A^*$ such that $u \lesssim_n^1 w$ and $v \lesssim_n^1 w$ with $u \neq v$. Then there exists $w' \in A^*$ with $|w'| = |w|$, $w' \neq w$, and $w \sim_n w'$.*

Proof idea. Since $|u| = |v| = |w| - 1$, w must be some $w_0a_1w_1a_2w_2$ with $a_1, a_2 \in A$ such that $u = w_0a_1w_1w_2$ and $v = w_0w_1a_2w_2$. We claim that $w' \stackrel{\text{def}}{=} w_0w_1a_2a_2w_2$ witnesses the lemma. Since $u \neq v$, we have $a_1w_1 \neq w_1a_2$, and thus $w \neq w'$. There remains to show that $w \sim_n w'$: this is done by a standard case analysis, see Appendix H. ◀

In the rest of this section, we consider some $\sim_n$-class $T \subseteq A^*$. The populous layers of T propagate upwards:

▶ **Lemma 6.3.** *If $T \cap A^{=p}$ is populous, then $T \cap A^{=p+1}$ is populous too.*

Proof. Suppose that T contains two distinct words u and v of length p. Then there is some w with $u \lesssim_n w \gtrsim_n v$ (Lemma 2.1.2) hence some u', v' with $u \lesssim_n^1 u'$ and $v \lesssim_n^1 v'$ (Lemma 2.1.1). If $u' \neq v'$ we are done since $|u'| = |v'| = p + 1$. If $u' = v'$ then $u \lesssim_n^1 u' \gtrsim_n^1 v$ and Lemma 6.2 shows that T contains at least two words of length $p + 1$. ◀

Populous layers also propagate downwards in the following sense:

▶ **Lemma 6.4.** *Let p be the length of the shortest word in T and suppose that $T \cap A^{=q}$ is populous, for some $q > p$. Then $T \cap A^{=p+1}$ is populous.*

Proof. Let q be the smallest layer such that $T \cap A^{=q}$ is populous. If $q = p + 1$ we are done, and similarly if $q = p$ (Lemma 6.3). So assume $q \geqslant p + 2$. For all ℓ with $p \leqslant \ell < q$, the layers $T \cap A^{=\ell}$ are nonempty (by Lemma 2.1) hence singular. Further, Lemma 2.1 tells us the form of the words in these layers: $T \cap A^{<q} = \{uv, uav, \ldots, ua^{q-p-1}v\}$ for some $u, v \in A^*$ and $a \in A$.

We now turn to $T \cap A^{=q}$. This populous layer contains some word $w \neq ua^{q-p}v$. By Theorem 6.2.9 of [27], all minimal (with respect to $\sqsubseteq$) words of T have the same length, hence w is not minimal in T, and is obtained by inserting a single letter in $ua^{q-p-1}v$. Define a word s as follows, depending on w:

- If $w = u'a^{q-p-1}v$ with $u' \sqsupseteq u$ and $|u'| = |u| + 1$, then $s = u'v$.
- If $w = u'a^\ell ba^{q-p-1-\ell}v$ with $b \neq a$, then $s = ubv$.
- If $w = ua^{q-p-1}v'$ with $v' \sqsupseteq v$ and $|v'| = |v| + 1$, then $s = uv'$.

The idea is that s is obtained by adding a letter to uv "exactly like" w is obtained from ua^{q-p-1v}. Since $w \neq ua^{q-p}v$, it is easy to see that $s \neq uav$. Since $uv \sqsubseteq s \sqsubseteq w$ and $uv \sim_n w$, we have $uv \sim_n s \sim_n w$. Thus T has at least two words of length $p + 1$, namely uav and s. ◀

We now handle a special case:

▶ **Lemma 6.5.** *If T is not linearly ordered by $\sqsubseteq$, then $C(T)$ is finite, and is in fact a subset of $A^{\leqslant m}$.*

Proof. Assume T is not linearly ordered by $\sqsubseteq$ and pick $u, v \in T$ with $u \not\sqsubseteq v$ and $|u| \leqslant |v|$. Let $q \stackrel{\text{def}}{=} |v|$. By Lemma 2.1, there exists $w \in T$ such that $u \lesssim_n w$ and $v \lesssim_n w$. By Lemma 2.1.2, there exists a $v' \in A^{=q}$ with $u \lesssim_n v' \lesssim_n w$. Furthermore, $v' \neq v$ since $u \not\sqsubseteq v$ and $u \sqsubseteq v'$. Thus $T \cap A^{=q}$ is populous. Since by the small-subword theorem the shortest word in T has length at most m, we conclude by Lemmas 6.4 and 6.3 that $T \cap A^{=p}$ is populous for every $p > m$. Thus $C(T) \subseteq A^{\leqslant m}$. ◀

We now consider the general case:

▶ **Lemma 6.6.** *$I(T)$ is $(m+1)$-PT.*

Proof. Recall that T is a singleton or is infinite (Lemma 2.1.4). We consider three cases.

- Suppose T is a singleton, $T = \{u\}$. By the small-subword theorem, $|u| \leqslant m$. Then $\uparrow u$ is m-PT, and $\downarrow u$ is $(m+1)$-PT. Thus $I(u) = A^* \smallsetminus (\uparrow u \cup \downarrow u)$ is $(m+1)$-PT.
- Suppose T is not a total order under $\sqsubseteq$. Then by Lemma 6.5, $C(T) \subseteq A^{\leqslant m}$, so $C(T)$ is $(m+1)$-PT, and so is $I(T)$.
- Suppose T is infinite and a total order under $\sqsubseteq$. Let p be the length of the shortest word in T. By the small-subword theorem, $p \leqslant m$. Since T is infinite, and by Lemma 2.1.2, $T \cap A^{=q}$ is nonempty for every $q \geqslant p$. Since T is a total order under $\sqsubseteq$, none of these $T \cap A^{=q}$ is populous, hence they are all singular. Therefore $C(T) \cap A^{\geqslant p} = T$. It remains to describe $C(T) \cap A^{\leqslant p}$, and this is $\downarrow u_0$, where u_0 is the unique word of length p in T. Thus $C(T) = T \cup \downarrow u_0$ is $(p+1)$-PT, hence also $(m+1)$-PT, and $I(T)$ too is $(m+1)$-PT. ◀

We may now conclude:

Proof of Theorem 6.1. Being n-PT, L is a finite union $T_1 \cup \cdots \cup T_\ell$ of equivalence classes of $\sim_n$, so that $I(L) = I(T_1) \cup \cdots \cup I(T_\ell)$. Now each $I(T_i)$ is $(m+1)$-PT by Lemma 6.6 so that $I(L)$ is too. ◀

▶ Remark 6.7. The upper bound in Theorem 6.1 is quite good: for any $k, \eta \geqslant 1$, the language $L = \{U_k\}$ from Proposition 3.1 has $h(U_k) = n = k\eta + 1$ so that Theorem 6.1 gives $h(I(U_k)) \leqslant (\eta+2)^k$. On the other hand we know by Eq. (11) that $h(I(U_k)) = |U_k| = (\eta+1)^k - 1$ when $k > 1$.

7 Deciding the two-variable logic of subwords

We assume familiarity with basic notions of first-order logic as exposed in, e.g., [7]: bound and free occurrences of variables, quantifier depth of formulae, and fragments FO^n where at most n different variables (free or bound) are used.

The signature of the $\mathsf{FO}(A^*, \sqsubseteq)$ logic consists of only one predicate symbol "$\sqsubseteq$", denoting the subword relation. Terms are variables taken from a countable set $X = \{x, y, z, \ldots\}$ and all words $u \in A^*$ as constant symbols (denoting themselves). For example, with $A = \{a, b, c, \ldots\}$, $\exists x\big(ab \sqsubseteq x \wedge bc \sqsubseteq x \wedge \neg(abc \sqsubseteq x)\big)$ is a true sentence as witnessed by $x \mapsto bcab$.

On motivations. Logics of sequences usually do not include the subsequence predicate and rather consider the prefix ordering, and/or functions for taking contiguous subsequences or computing the length of sequences, see, e.g., [5, 10]. However, in automated deduction, and specifically in ordered constraints solving, the decidability of logics of simplification orderings on strings and trees – $\mathsf{FO}(A^*, \sqsubseteq)$ being a special case – is a key issue [3, 19]. These works often limit their scope to Σ_1 or similar fragments since decidability is elusive in this area.

7.1 Decidability for $\mathsf{FO}^2(A^*, \sqsubseteq)$

In [14] we showed that validity and satisfiability are decidable for the FO^2 fragment of the logic of subwords (note that the $\mathsf{FO}^3 \cap \Sigma_2$ fragment is undecidable [19, 14]). Since below we use our results on the heights of PT languages to prove a new complexity upper bound on the underlying algorithm, we first need to recall the main lines of the decidability proof (see [14] for full details).

When describing the decision procedure for the FO^2 fragment, it is convenient to enrich the basic logic by allowing all regular expressions as monadic predicates (with the expected

semantics) and we shall temporarily adopt this extension. For example, we can state that the downward closure of $(ab)^*$ is exactly $(a+b)^*$ with $\forall x\big[x \in (a+b)^* \iff \exists y(y \in (ab)^* \wedge x \sqsubseteq y)\big]$.

In the following we consider FO^2 formulae using only x and y as variables. We consider a variant of the logic where we use the binary relations $\sqsubset$, $\sqsupset$, $=$ and $\perp$ instead of $\sqsubseteq$. This will be convenient later. The two variants are equivalent, even when restricting to FO^m fragments, since the new set of predicates can be defined in terms of $\sqsubseteq$ and vice versa. To simplify notation, we sometimes use negated predicate symbols as in $x \not\sqsubseteq y$ or $x \notin (ab)^*$ with obvious meaning.

▶ **Lemma 7.1.** *Let $\phi(x)$ be an $\mathsf{FO}^2(A^*, \sqsubseteq)$ formula with at most one free variable. Then there exists a regular language $L_\phi \subseteq A^*$ such that $\phi(x)$ is equivalent to $x \in L_\phi$. Furthermore, L_ϕ can be built effectively from ϕ and A.*

Proof. By structural induction on $\phi(x)$. If $\phi(x)$ is an atomic formula of the form $x \in L$, the result is immediate. If $\phi(x)$ is an atomic formula that uses a binary predicate, the fact that it has only one free variable means that $\phi(x)$ is a trivial $x = x$, $x \sqsubset x$, $x \sqsupset x$ or $x \perp x$, so that L_ϕ is A^* or $\varnothing$.

For formulae of the form $\neg\phi'(x)$ or $\phi_1(x) \vee \phi_2(x)$, we use the induction hypothesis and the fact that regular languages are (effectively) closed under boolean operations.

The remaining case is when $\phi(x)$ has the form $\exists y\, \phi'(x, y)$. Using the induction hypothesis, we replace any subformulae of ϕ' having the form $\exists x\, \psi(x, y)$ or $\exists y\, \psi(x, y)$ with equivalent formulae of the form $y \in L_\psi$ or $x \in L_\psi$ respectively, for appropriate languages L_ψ. Now ϕ' is quantifier-free. We further rewrite it by pushing all negations inside with the following meaning-preserving rules:

$$\neg(\psi_1 \vee \psi_2) \to \neg\psi_1 \wedge \neg\psi_2\,, \qquad \neg(\psi_1 \wedge \psi_2) \to \neg\psi_1 \vee \neg\psi_2\,, \qquad \neg\neg\psi \to \psi\,,$$

and then eliminating negations completely with:

$$\neg(z \in L) \to z \in (A^* \smallsetminus L)\,, \qquad \neg(z_1\ R_1\ z_2) \to z_1\ R_2\ z_2 \vee z_1\ R_3\ z_2 \vee z_1\ R_4\ z_2\,,$$

where R_1, R_2, R_3, R_4 is any permutation of $\mathcal{R} \stackrel{\text{def}}{=} \{=, \sqsubset, \sqsupset, \perp\}$. This last rewrite rule is correct since the four relations form a partition of $A^* \times A^*$: for all $u, v \in A^*$, exactly one of $u = v$, $u \sqsubset v$, $u \sqsupset v$, and $u \perp v$ holds.

Thus, we may now assume that ϕ' is a positive boolean combination of atomic formulae. We write ϕ' in disjunctive normal form, that is, as a disjunction of conjunctions of atomic formulae. Observing that $\exists y(\phi_1 \vee \phi_2)$ is equivalent to $\exists y\, \phi_1 \vee \exists y\, \phi_2$, we assume w.l.o.g. that ϕ' is just a conjunction of atomic formulae. Any atomic formula of the form $x \in L$, for some L, can be moved outside the existential quantification, since $\exists y(x \in L \wedge \psi)$ is equivalent to $x \in L \wedge \exists y\, \psi$. All atomic formulae of the form $y \in L$ can be combined into a single one, since regular languages are closed under intersection.

Finally we may assume that $\phi'(x, y)$ is a conjunction of a single atomic formula of the form $y \in L$ (if no such formula appears, we can write $y \in A^*$), and some combination of atomic formulae among $x \sqsubset y$, $x \sqsupset y$, $x = y$, and $x \perp y$. If at least two of these appear, then their conjunction is unsatisfiable, and so $\phi(x)$ is equivalent to $x \in \varnothing$. If none of them appear, $\exists y(y \in L)$ is equivalent to $x \in A^*$ (or to $x \in \varnothing$ if L is empty). If exactly one of them appears, say $x\ R\ y$, then $\exists y\ (y \in L \wedge xRy))$ is equivalent to $x \in L_\phi$ for $L_\phi = R^{-1}(L)$. Now the pre-image $R^{-1}(L)$ is regular and effectively computable from L since all the relations in $\mathcal{R}$ are rational relations.[7] ◀

[7] This is well known and easy to see for $\sqsubset$ and $\sqsupset$. It is proved in [14] for $\perp$.

▶ **Corollary 7.2** ([14]). *The truth problem for* $\mathsf{FO}^2(A^*, \sqsubseteq)$ *is decidable.*

Proof. Lemma 7.1 provides a recursive procedure for computing L_ϕ, the set of words that make $\phi(x)$ true. When ϕ is a closed formula, it is true iff L_ϕ is A^*. ◀

Complexity for $\mathsf{FO}^2(A^*, \sqsubseteq)$. The algorithm underlying the proof of Lemma 7.1 can be implemented using finite-state automata to handle the regular languages L_ϕ that are constructed for each subformula. However, steps like complementation or even computing the pre-images $\uparrow_< L$ and $\downarrow_< L$ are costly and may incur an exponential blowup, and this cannot be avoided by using nondeterministic or alternating automata instead of standard deterministic automata [12]. The consequence is that the only clear upper bound for the algorithm is a tower of exponentials whose height is given by the quantifier depth of the formula at hand, hence a nonelementary complexity. Regarding lower bounds, only PSPACE-hardness has been established [14] and we conjecture that $\mathsf{FO}^2(A^*, \sqsubseteq)$ can be decided with elementary complexity.

7.2 Complexity of the $\mathsf{FO}^2(A^*, \sqsubseteq)$ logic without regular predicates

It turns out that when regular predicates are not allowed (i.e., when we use the basic logic), the quantifier-elimination procedure will only produce membership constraints $x \in L$ or $y \in L'$ involving PT languages. Furthermore, it is possible to bound the PT height of the defined languages and deduce an elementary complexity upper bound.

▶ **Theorem 7.3** ($\mathsf{FO}^2(A^*, \sqsubseteq)$ has elementary complexity). *If $\phi(x)$ is an FO^2 formula without regular predicates, then L_ϕ is a piecewise-testable language with $h(L_\phi)$ in $2^{2^{O(|\phi|)}}$.*

Furthermore, computing a canonical DFA for L_ϕ (hence deciding the truth of ϕ) can be done in 3−EXPTIME.

Proof. We mimic the proof of Lemma 7.1. In this process we can allow atomic formulae "$x \in L$" when L is PT, since this can be expressed as a boolean combination of atomic formulae of the form $w \sqsubseteq x$. The key extra ingredient is that the pre-images $R^{-1}(L)$ preserve piecewise-testability and that $h(R^{-1}(L))$ is in $O(h(L)^{|A|})$: we invoke Theorem 4.4 for $R = \sqsupset$, Theorem 5.4 for $R = \sqsubset$, and Theorem 6.1 for $R = \perp$.

Finally, when the PT height of L_ϕ (and of all intermediary L_ψ) have been bounded in $2^{2^{O(|\phi|)}}$, we obtain a bound on the size of the DFAs and the time and space needed to compute them using Theorem 2.2. ◀

8 Concluding remarks

We developed several new techniques for proving upper and lower bounds on the PT height of languages constructed by closing w.r.t. the subword ordering or its inverse. We also considered related constructions like taking minimal elements, or taking the image by the incomparability relation. In general, the PT height of upward closures is bounded with the length of minimal words. For downward closures, we developed techniques for expressing them with D-products and bounding their lengths. We illustrated these techniques with regular and context-free languages but more classes can be considered [31]. More importantly, the closures of PT languages have PT height bounded polynomially in terms of the PT height of the argument. Our main tool here is the small-subword theorem that provides tight lower bounds on the PT height of finite languages, with ad hoc developments for $I(L)$.

These results are used to bound the complexity of the two-variable logic of subwords but we believe that the PT hierarchy can be used more generally as an effective measure of descriptive complexity. (The same can be said of the hierarchies of locally-testable languages, or of dot-depth-one languages).

This research program raises many interesting questions, such as connecting PT height and other measures, narrowing the gaps remaining in our Theorems 4.4, 5.4, and 6.1, and enriching the known collection of PT-preserving operations.

These questions will probably require new insights in PT languages. For example, the experiments we conducted suggest that $h(u \sqcup\!\sqcup v)$ can be bounded by $h(u) + h(v)$ when u, v are words, however we do not know how to prove this. Similarly, we can prove that $L \sqcup\!\sqcup u$ is PT when L is PT and u is a word, but we only have a very tedious proof. We mention these questions since $L \sqcup\!\sqcup A$ is the pre-image of L by $\trianglerighteq^1$ and it seems that the decidability of $\mathsf{FO}^2(A^*, \sqsubseteq)$ can be extended to $\mathsf{FO}^2(A^*, \sqsubseteq, \sqsubseteq^1)$ [20].

References

1 G. Bachmeier, M. Luttenberger, and M. Schlund. Finite automata for the sub- and superword closure of CFLs: Descriptional and computational complexity. In *Proc. LATA 2015*, volume 8977 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2015. doi:10.1007/978-3-319-15579-1_37.

2 M. Bojańczyk, L. Segoufin, and H. Straubing. Piecewise testable tree languages. *Logical Methods in Comp. Science*, 8(3), 2012. doi:10.2168/LMCS-8(3:26)2012.

3 H. Comon. Solving symbolic ordering constraints. *Int. J. Foundations of Computer Science*, 1(4):387–412, 1990. doi:10.1142/S0129054190000278.

4 V. Diekert, P. Gastin, and M. Kufleitner. A survey on small fragments of first-order logic over finite words. *Int. J. Foundations of Computer Science*, 19(3):513–548, 2008. doi:10.1142/S0129054108005802.

5 V. Ganesh, M. Minnes, A. Solar-Lezama, and M. C. Rinard. Word equations with length constraints: What's decidable? In *Proc. HVC 2012*, volume 7857 of *Lecture Notes in Computer Science*, pages 209–226. Springer, 2013. doi:10.1007/978-3-642-39611-3_21.

6 L. H. Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1):94–98, 1969. doi:10.1016/S0021-9800(69)80111-0.

7 J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009. doi:10.1017/CBO9780511576430.

8 G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc. (3)*, 2(7):326–336, 1952. doi:10.1112/plms/s3-2.1.326.

9 P. Hofman and W. Martens. Separability by short subsequences and subwords. In *Proc. ICDT 2015*, volume 31 of *Leibniz International Proceedings in Informatics*, pages 230–246, 2015. doi:10.4230/LIPIcs.ICDT.2015.230.

10 P. Hooimeijer and W. Weimer. StrSolve: solving string constraints lazily. *Autom. Softw. Eng.*, 19(4):531–559, 2012. doi:10.1007/s10515-012-0111-x.

11 P. Karandikar, M. Kufleitner, and Ph. Schnoebelen. On the index of Simon's congruence for piecewise testability. *Information Processing Letters*, 115(4):515–519, 2015. doi:10.1016/j.ipl.2014.11.008.

12 P. Karandikar, M. Niewerth, and Ph. Schnoebelen. On the state complexity of closures and interiors of regular languages with subwords and superwords. *Theoretical Computer Science*, 610:91–107, 2016. doi:10.1016/j.tcs.2015.09.028.

13 P. Karandikar and Ph. Schnoebelen. On the index of Simon's congruence for piecewise testability (v2), October 2013. This version available at arxiv:1310.1278v2. URL: http://arxiv.org/abs/1310.1278v2.

14 P. Karandikar and Ph. Schnoebelen. Decidability in the logic of subsequences and supersequences. In *Proc. FST&TCS 2015*, volume 45 of *Leibniz International Proceedings in Informatics*, pages 84–97, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.84.

15 O. Klíma. Piecewise testable languages via combinatorics on words. *Discrete Mathematics*, 311(20):2124–2127, 2011. doi:10.1016/j.disc.2011.06.013.

16 O. Klíma and L. Polák. Alternative automata characterization of piecewise testable languages. In *Proc. DLT 2013*, volume 7907 of *Lecture Notes in Computer Science*, pages 289–300. Springer, 2013. doi:10.1007/978-3-642-38771-5_26.

17 L. Kontorovich, C. Cortes, and M. Mohri. Kernel methods for learning languages. *Theoretical Computer Science*, 405(3):223–236, 2008. doi:10.1016/j.tcs.2008.06.037.

18 J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297–305, 1972. doi:10.1016/0097-3165(72)90063-5.

19 D. Kuske. Theories of orders on the set of words. *RAIRO Theoretical Informatics and Applications*, 40(1):53–74, 2006. doi:10.1051/ita:2005039.

20 D. Kuske. Private communication, September 2015.

21 T. Masopust. Piecewise testable languages and nondeterministic automata. arXiv:1603.00361 [cs.FL], March 2016. URL: http://arxiv.org/abs/1603.00361.

22 T. Masopust and M. Thomazo. On the complexity of k-piecewise testability and the depth of automata. In *Proc. DLT 2015*, volume 9168 of *Lecture Notes in Computer Science*, pages 364–376. Springer, 2015. doi:10.1007/978-3-319-21500-6_29.

23 O. Matz. On piecewise testable, starfree, and recognizable picture languages. In *Proc. FOSSACS'98*, volume 1378 of *Lecture Notes in Computer Science*, pages 203–210. Springer, 1998. doi:10.1007/BFb0053551.

24 D. Perrin and J.-É. Pin. *Infinite words: Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics Series*. Elsevier Science, 2004.

25 J.-É. Pin. *Varieties of Formal Languages*. Plenum, New-York, 1986.

26 J. Rogers, J. Heinz, M. Fero, J. Hurst, D. Lambert, and S. Wibel. Cognitive and sub-regular complexity. In *Proc. FG 2012 & 2013*, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013. doi:10.1007/978-3-642-39998-5_6.

27 J. Sakarovitch and I. Simon. Subwords. In M. Lothaire, editor, *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*, chapter 6, pages 105–142. Cambridge Univ. Press, 1983.

28 A. Salomaa, D. Wood, and Sheng Yu. On the state complexity of reversals of regular languages. *Theoretical Computer Science*, 320(2–3):315–329, 2004. doi:10.1016/j.tcs.2004.02.032.

29 I. Simon. Piecewise testable events. In *Proc. 2nd GI Conf. on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975. doi:10.1007/3-540-07407-4_23.

30 I. Simon. Words distinguished by their subwords. In *Proc. WORDS 2003*, 2003.

31 G. Zetzsche. An approach to computing downward closures. In *Proc. ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015. doi:10.1007/978-3-662-47666-6_35.

A Operations that do not preserve piecewise-testability

In this appendix we give examples showing that PT languages are not closed under many usual language-theoretic operations.

Concatenation. $a(a+b)^*$ is not PT: it contains $(ab)^k$ but not $b(ab)^k$, however $(ab)^k \sim_k b(ab)^k$ for any k. Hence the class PT is not closed under concatenation (even in the special case of prefixing with a) since $(a+b)^*$ is PT.

Kleene star. PTis not closed under Kleene star (recall that PTis a subvariety of the star-free languages): aa is finite hence PT but $(aa)^*$ is not PT.

Shuffle product. ab^* and a^* are PT but their shuffle product $ab^* \sqcup\!\sqcup a^* = a(a+b)^*$ is not.

Conjugacy. Recall that the conjugates of u are $\widetilde{u} \stackrel{\text{def}}{=} \{u_2u_1 \mid u = u_1u_2\}$, and we extend with $\widetilde{L} = \bigcup_{u\in L} \widetilde{u}$. Now $ac(a+b)^*$ is PT but $\widetilde{ac(a+b)^*} = (a+b)^*ac(a+b)^* + c(a+b)^*a$ is not.

Renaming. $c(a+b)^*$ is PT but applying the renaming $c \mapsto a$ yield $a(a+b)^*$.

Erasing one letter. This operation can be seen as the inverse of $L \mapsto L \sqcup\!\sqcup A$ where an arbitrary letter is inserted at an arbitrary position. Now $ac(a+b)^*$ is PT but erasing one letter yields $(a+c+ac)(a+b)^*$ which is not PT.

B Image of PT-languages by inverse morphisms

For the sake of completeness, we give a proof of Eq. (4) from page 4:

▶ **Lemma B.1.** *Let $\rho : A^* \to B^*$ be a monoid morphism. If $L \subseteq B^*$ is n-PT then $\rho^{-1}(L) \subseteq A^*$ is also n-PT.*

Proof. Consider a word $w \in B^*$ of length p with $p \leqslant n$. We start by showing that $\rho^{-1}(\uparrow w)$ is n-PT. Clearly $\rho^{-1}(\uparrow w)$ is upward-closed and hence PT, it remains to be shown that all minimal elements of $\rho^{-1}(\uparrow w)$ are of length at most n. Let v be a minimal element of $\rho^{-1}(\uparrow w)$. Then $w \sqsubseteq \rho(v)$. Write $w = w_1 \dots w_p$ with each w_i a letter, and v as $v_1 \dots v_q$ with each v_j a letter. We have $w_1 \dots w_p \sqsubseteq \rho(v_1) \dots \rho(v_q)$. Further, by minimality of v, if any factor $\rho(v_j)$ is removed from the right hand side, the relation will no longer hold. Thus $q \leqslant p$ and so $q \leqslant n$. Since $\rho^{-1}(\uparrow w)$ is a union of sets of the form $\uparrow v$ with $|v| \leqslant n$, $\rho^{-1}(\uparrow w)$ is n-PT.

Finally, note that inverse morphisms preserve boolean operations, that is, $\rho^{-1}(B^* \setminus L) = A^* \setminus \rho^{-1}(L)$, and $\rho^{-1}(L_1 \cup L_2) = \rho^{-1}(L_1) \cup \rho^{-1}(L_2)$. Every n-PT language $L \subseteq B^*$ is a boolean combination of principal filters $\uparrow w$ with $|w| \leqslant n$, and so the result follows. ◀

C A polynomial-time algorithm for the PT height of single words.

It is not too hard to compute the PT height of a singleton language as we now explain. For words $u, v \in A^*$, let $\delta(u,v) = \min\{n : u \not\sim_n v\}$ if $u \neq v$ and $\delta(u,v) = \infty$ if $u = v$.

Let us first describe a simple algorithm to compute $\delta(u,v)$, given u and v. Assume $u \neq v$. Then $\delta(u,v)$ is the smallest length n such that some word of length n is a subword of exactly one of u and v. For any word w, the canonical complete DFA $\mathcal{A}_{\downarrow w}$ for the set of all subwords of w has $|w|+2$ states and is easy to build. Then $\delta(u,v)$ is the length of a shortest word in $L(\mathcal{A}_{\downarrow u}) \mathbin{\triangle} L(\mathcal{A}_{\downarrow v})$, where $\triangle$ denotes symmetric difference. Using the product construction for DFAs, one can compute $\delta(u,v)$ in time $O(|u| \cdot |v| \cdot |A|)$. A more involved algorithm to compute $\delta(u,v)$ in time $O(|u| + |v| + |A|)$ is presented in [30].

Note that $h(u)$ is the smallest n such that the equivalence class of u under $\sim_n$ is just $\{u\}$. If $[u]_n$ is not a singleton, then it has infinitely many elements (see Lemma 2.1.4), in particular, some word of length greater than $|u|$, and therefore some word v such that $u \sqsubseteq v$ and $|v| = |u| + 1$ (see Lemma 2.1.2). The number of such words v is at most $(|u|+1) \cdot |A|$, and so computing $\delta(u,v)$ for all such v allows us to compute $h(u)$:

$$h(u) = \max_{v \in u \sqcup\!\sqcup A} \delta(u,v)\,.$$

This gives an overall time complexity upper bound of $O(|u|^3 \cdot |A|^2)$. Using the algorithm from [30] to compute δ, this can be improved to $O((|u| + |A|) \cdot |u| \cdot |A|)$.

D Proof that P_k and N_k characterize U_k

▶ **Claim.** *For any $k \in \mathbb{N}$ and $u \in A_k^*$:*

$$\Big(\bigwedge_{v \in P_k} v \sqsubseteq u\Big) \wedge \Big(\bigwedge_{w \in N_k} w \not\sqsubseteq u\Big) \iff u = U_k. \tag{6}$$

Proof. By induction on k. For $k = 0$, A_0 is empty and there is only one word in A_0^*, namely $u = U_0 = \varepsilon$. It satisfies the positive constraint $U_0 \sqsubseteq u$ and there are no negative constraints in N_0.

Assume now that $k > 0$ and that the claim holds for $k - 1$. We prove the left-to-right implication: Since P_k is not empty, the P_k constraints $a_k^i v a_k^{\eta - i} \sqsubseteq u$ imply that $|u|_{a_k} \geqslant \eta$. However the N_k constraint $a_k^{\eta+1} \not\sqsubseteq u$ implies that u contains exactly η occurrences of a_k and can be written $u = v_0 a_k v_1 a_k \cdots a_k v_\eta$ with $v_i \in A_{k-1}^*$ for all $i = 0, \ldots, \eta$.

Consider some fixed v_i: for any $v \in P_{k-1}$ it holds that $v \sqsubseteq v_i$ since $a_k^i v a_k^{\eta - i} \sqsubseteq u$. Similarly $w \not\sqsubseteq v_i$ for any $w \in N_{k-1}$ since $a_k^i w a_k^{\eta - i} \not\sqsubseteq u$. The ind. hyp. now yields $v_i = U_{k-1}$, thus $u = U_{k-1} a_k U_{k-1} \cdots a_k U_{k-1} = U_k$. The right-to-left implication should now be clear and can be left to the reader. ◀

E Computing $h(\downarrow U_k)$

Let $U_0 = \varepsilon$ and, for $k > 0$, $U_k = (U_{k-1} a_k)^\eta U_k$. Write L_k for $|U_k|_{a_1}$ and note that $L_k = (\eta + 1) L_{k-1}$ when $k > 1$.

▶ **Claim.** *For any $k, r \in \mathbb{N}$, if $x \sqsubseteq U_k^r$, then $h(x) \leqslant 1 + r L_k$.*

Proof. By induction on k. For $k \leqslant 1$, $U_k^r = a_1^{r L_k}$ requires $x = a_1^\ell$ with $\ell \leqslant r L_k$ so $h(x) = 1 + \ell \leqslant 1 + r L_k$.

So assume $k > 1$. Let $m = |x|_{a_k}$ and factor x as $x_0 a_k x_1 a_k \ldots a_k x_m$ so that $x_i \in A_{k-1}^*$ for all i. Now, for any $y \in A^*$, the following holds:

$$y = x \iff a_k^m \sqsubseteq y \wedge a_k^{m+1} \not\sqsubseteq y \wedge \bigwedge_{i=0}^{m} \bigwedge_{u \in A^{\leqslant h(x_i)}} (a_k^i u a_k^{m-i} \sqsubseteq y \iff u \sqsubseteq x_i)\,. \tag{12}$$

We deduce that $h(x) \leqslant \max(m+1, m+h(x_0), \ldots, m+h(x_m)) = m + \max(1, h(x_0), \ldots, h(x_m))$. Note that $x_i \sqsubseteq U_{k-1}^{r'}$ for $r' = r(\eta+1) - m$, so, by induction hypothesis, $h(x_i) \leqslant 1 + r' L_{k-1}$. Assuming $k > 1$, we thus have

$$\begin{aligned} h(x) \leqslant m + 1 + r' L_{k-1} &= m + 1 + [r(\eta+1) - m] L_{k-1} \\ &= 1 + m[1 - L_{k-1}] + r(\eta+1) L_{k-1} \leqslant 1 + r L_k\,. \end{aligned}$$

◀

▶ **Corollary E.1.** *$h(\downarrow U_k^r) = 1 + r L_k$, and thus in particular, $h(\downarrow U_k) = 1 + L_k$.*

Proof. We use Eq. (5) and note that $a_1^{r L_k} \in \downarrow U_k^r$. Hence $h(\downarrow U_k^r) \geqslant h(a_1^{r L_k}) = 1 + r L_k$. ◀

F Proofs for Lemmas 3.2 and 3.3

▶ **Lemma 3.2** (restated). *If u_1 and u_2 are respectively ℓ_1-rich and ℓ_2-rich, then $v \sim_n v'$ implies $u_1vu_2 \sim_{\ell_1+n+\ell_2} u_1v'u_2$.*

Proof. A subword x of u_1vu_2 can be decomposed as $x = x_1yx_2$ where x_1 is the longest prefix of x that is a subword of u_1 and x_2 is the longest suffix of the remaining $x_1^{-1}x$ that is a subword of u_2. Thus $y \sqsubseteq v$ since $x \sqsubseteq u_1vu_2$. Now, since u_1 is ℓ_1-rich, $|x_1| \geqslant \ell_1$ (unless x is too short), and similarly $|x_2| \geqslant \ell_2$ (unless ...). Finally $|y| \leqslant n$ when $|x| \leqslant \ell_1 + n + \ell_2$, and then $y \sqsubseteq v'$ since $v \sim_n v'$, entailing $x \sqsubseteq u_1v'u_2$. A symmetrical reasoning shows that subwords of $u_1v'u_2$ of length $\leqslant \ell_1 + n + \ell_2$ are subwords of u_1vu_2 and we are done. ◀

▶ **Lemma 3.3** (restated). *Consider two words u, u' of richness m and with rich factorizations $u = u_1a_1 \cdots u_ma_my$ and $u' = u'_1a_1 \cdots u'_ma_mv'$. Suppose that $v \sim_n v'$ and that $u_i \sim_{n+1} u'_i$ for all $i = 1, \ldots, m$. Then $u \sim_{n+m} u'$.*

Proof. By repeatedly using Lemma 3.2, one shows

$$\begin{aligned} u_1a_1u_2a_2 \cdots u_ma_mv &\sim_{n+m} u'_1a_1u_2a_2 \cdots u_ma_mv \\ &\sim_{n+m} u'_1a_1u'_2a_2 \cdots u_ma_mv \\ &\vdots \\ &\sim_{n+m} u'_1a_1u'_2a_2 \cdots u'_ma_mv \\ &\sim_{n+m} u'_1a_1u'_2a_2 \cdots u'_ma_mv' \,, \end{aligned}$$

using the fact that each factor u_ia_i is rich. ◀

G Proofs for Theorem 5.4

▶ **Lemma G.1.** *If $uLv \subseteq [uv]_n$, where $L \subseteq A^*$ is any language, then $u(\downarrow L)v \subseteq [uv]_n$.*

Proof sketch. Recall that $w_1 \sim_n w_2$ and $w_1 \sqsubseteq w_2$ implies $w_1 \sim_n w'$ for all $w_1 \sqsubseteq w' \sqsubseteq w_2$. ◀

▶ **Lemma 5.5** (restated). *If $uB^*C^*B^*v \subseteq [uv]_n$, where $B, C \subseteq A$ are subalphabets, then $u(B \cup C)^*v \subseteq [uv]_n$.*

Proof. We prove that for any $m \in \mathbb{N}$, for any $w \in B^*(C^*B^*)^m$, for any $s \in A^{\leqslant n}$, $s \sqsubseteq uwv$ implies $s \sqsubseteq uv$. The proof is by induction on m, knowing that the claim holds by assumption for $m \leqslant 1$.

Assume therefore that $m \geqslant 2$ and write w as $w = xyz$ with $x \in B^*C^*$, $y \in B^*(C^*B^*)^{m-2}$, and $z \in C^*B^*$. If $s \sqsubseteq uwv = uxyzv$ then s can be factored as $s = s_us_xs_ys_zs_v$ with each factor s_* a subword of the corresponding factor of uwv. Let $s' \stackrel{\text{def}}{=} s_us_xs_zs_v$ so that $s' \sqsubseteq uxzv$. Note that $xz \in B^*C^*B^*$ hence $s' \sqsubseteq uxzv$ entails $s' \sqsubseteq uv$ by assumption. Thus either $s_us_x \sqsubseteq u$ or $s_zs_v \sqsubseteq v$.

In the first case, $s = s_us_xs_ys_zs_v \sqsubseteq uyzv$ and since $yz \in B^*(C^*B^*)^{m-1}$ the induction hypothesis applies and yields $s \sqsubseteq uv$.

In the second case a symmetrical reasoning applies. ◀

▶ **Lemma G.2.** *If $uB^*C^*LD^*B^*v \subseteq [uv]_n$, where $B, C, D \subseteq A$ are subalphabets and $L \subseteq A^*$ is any language then $u(B \cup C)^*L(B \cup D)^*v \subseteq [uv]_n$.*

Proof. We assume that $L \neq \varnothing$ (otherwise the result holds trivially) so that $uB^*C^*LD^*B^*v \subseteq [uv]_n$ entails $uB^*C^*B^*v \subseteq [uv]_n$ (by Lemma G.1), hence $u(B \cup C)^*v \subseteq [uv]_n$ (by Lemma 5.5).

We now prove that for $s \in A^{\leqslant n}$ and $w \in (B^*C^*)^k L(D^*B^*)^\ell$, $s \sqsubseteq uwv$ implies $s \sqsubseteq uv$. The proof is by induction on $k + \ell \in \mathbb{N}$. Note that the Lemma's assumption handles all cases with $k \leqslant 1$ and $\ell \leqslant 1$.

Let us therefore assume $k > 1$ since the case where $\ell > 1$ is symmetrical. Assume $s \sqsubseteq uwv$ and write w as $w = xyz$ with $x \in B^*C^*$, $y \in (B^*C^*)^{k-1}$, and $z \in L(D^*B^*)^\ell$.

Since $s \sqsubseteq uwv = uxyzv$ there is a factorization $s = s_u s_x s_y s_z s_v$ of s with each factor s_* embedding in the corresponding factor of $uxyzv$. Let now $s' \stackrel{\text{def}}{=} s_u s_x s_z s_v$: this word satisfies $s' \sqsubseteq uxzv$ and $|s'| \leqslant n$. Now $uxzv \in uB^*C^*L(D^*B^*)^\ell v$, so that we may apply the induction hypothesis and deduce $s' \sqsubseteq uv$ from $s' \sqsubseteq uxzv$. Thus either $s_u s_x \sqsubseteq u$ or $s_z s_v \sqsubseteq v$.

If $s_u s_x \sqsubseteq u$ we deduce

$$s = s_u s_x s_y s_z s_v \sqsubseteq uyzv. \tag{13}$$

Now $yz \in (B^*C^*)^{k-1}L(D^*B^*)^\ell$ so that we can apply the induction hypothesis and deduce $s \sqsubseteq uv$ from Eq. (13).

If $s_z s_v \sqsubseteq v$ we deduce

$$s = s_u s_x s_y s_z s_v \sqsubseteq uxyv. \tag{14}$$

Now $xy \in (B^*C^*)^k$ so that $uxyv \in [uv]_n$ as we observed at the beginning. Thus from Eq. (14) we deduce $s \sqsubseteq uv$. ◀

We now give an application of the above lemma in a form which we will use later:

▶ **Lemma 5.6** (restated). *Suppose $L_1 B_1^* B_2^* \cdots B_\ell^* L_2 \subseteq [u]_n$ for some languages $L_1, L_2 \subseteq A^*$ and subalphabets $B_1, B_2, \ldots, B_\ell \subseteq A$ with $\ell \geqslant 3$. If $a \in B_1 \cap B_\ell$ then, letting $B'_i = B_i \cup \{a\}$, $L_1 B'^*_1 B'^*_2 \cdots B'^*_\ell L_2 \subseteq [u]_n$.*

Proof. By induction on ℓ. Write $L_1 B_1^* B_2^* \cdots B_\ell^* L_2$ as

$$L_1 B_1^*\ a^* B_2^* \cdots B_{\ell-1}^* a^*\ B_\ell^* L_2\ .$$

For every $u_1 \in L_1 B_1^*$ and $u_2 \in B_\ell^* L_2$, we have

$$u_1\ a^* B_2^* \cdots B_{\ell-1}^* a^* u_2 \subseteq [u]_n\ .$$

Lemma G.2 gives $u_1\ B'^*_2 B_3^* \cdots B_{\ell-2}^* B'^*_{\ell-1}\ u_2 \subseteq [u]_n$, hence $u_1\ B'^*_2 B'^*_3 \cdots B'^*_{\ell-2} B'^*_{\ell-1}\ u_2 \subseteq [u]_n$ by the induction hypothesis. Since this applies to all $u_1 \in L_1 B_1^* = L_1 B'^*_1$ and $u_2 \in B_2^* L_2 = B'^*_2 L_2$, we have proven the lemma. ◀

▶ **Lemma 5.8** (restated). *Let $L \subseteq A^*$ be n-PT. Let $k = |A|$ and $m = f_k(n)$. For every $u \in L$ there is a D-product P_u of length $\ell \leqslant mk + m + k$ such that $u \in P_u \subseteq [u]_n \subseteq L$.*

In the above statement (and below) we abuse notation and let P denote both a regular expression and the language (a subset of A^*) it denotes.

Proof. Assume $u \in L$. By the small-subword theorem, and since L is closed under $\sim_n$, there exists a subword $v = a_1 \ldots a_\ell$ of u with $v \sim_n u$ and $\ell = \ell \leqslant m$. Thus u has the form

$$u = b_{0,1} \cdots b_{0,p_0}\, a_1\, b_{1,1} \cdots b_{1,p_1}\, a_2 \cdots a_\ell\, b_{\ell,1} \cdots b_{\ell,p_\ell}\ .$$

Here the $b_{i,j}$'s are the letters from u that do not occur in the subword v. To shorten notation, we write $u = \prod_{i=0}^{\ell}(a_i \prod_{j=1}^{p_i} b_{i,j})$, abusing notation by letting $a_0 = \varepsilon$.

By the pumping property (Lemma 2.1.3), we deduce that $P \stackrel{\text{def}}{=} \prod_{i=0}^{\ell}(a_i \prod_{j=1}^{p_i}\{b_{i,j}\}^*)$ is a D-product satisfying $u \in P \subseteq [u]_n \subseteq L$.

We now modify this product to take advantage of the more general pumping property. Let $P' = \prod_{i=0}^{\ell}(a_i \prod_{j=1}^{p_i} B_{i,j}^*)$ where $B_{i,j} = \{b_{i,j}\} \cup \{a \in A \mid \exists 1 \leqslant j_1 < j < j_2 \leqslant p_i \ : \ a = b_{i,j_1} = b_{i,j_2}\}$. That is, every subalphabet $\{b_{i,j}\}$ in P is completed with any letter that appears both before and after $b_{i,j}$ in the same i-th segment $B_{i,1}^* \cdots B_{i,p_i}^*$. Now Lemma 5.6 ensures that $P \subseteq P' \subseteq [u]_n \subseteq L$ (and we still have $u \in P'$ since $P \subseteq P'$).

We now simplify P' by repeatedly replacing a factor $B^*B'^*$ where $B \subseteq B'$ (or $B' \subseteq B$) by B'^* (or B^*). This does not change the language denoted by P'. When no more simplifications are possible, we let $P_u \stackrel{\text{def}}{=} \prod_{i=0}^{\ell}(a_i \prod_{j=1}^{\ell_i} C_{i,j}^*)$ denote the simplified D-product. For any $i \in \{0, \ldots, \ell\}$, the sequence of sets $C_{i,1}, \ldots, C_{i,\ell_i}$ satisfies the hypothesis of Lemma 5.7, and thus $\ell_i \leqslant k = |A|$. This entails that P_u has length bounded by $(m+1)(k+1) - 1$ (recall that $a_0 = \varepsilon$), i.e. by $mk + m + k$. ◀

H Proof that $w \sim_n w'$ for Lemma 6.2

Recall that $u = w_0a_1w_1w_2$, $v = w_0w_1a_2w_2$, with $w = w_0a_1w_1a_2w_2$ and $w' = w_0w_1a_2a_2w_2$. Since $w \sim_n v \sqsubseteq w'$, we only have to show that any subword of length at most n of w' is also a subword of w.

So let $s \sqsubseteq w'$ with $|s| \leqslant n$. Factorize s as $s = v_0v_1s'v_2$ as follows:

1. Let v_0 be the longest prefix of s such that $v_0 \sqsubseteq w_0$.
2. Having fixed v_0, let v_1 be the longest prefix of $(v_0)^{-1}s$ such that $v_1 \sqsubseteq w_1$.
3. Having fixed v_0 and v_1, let v_2 be the longest suffix of $(v_0v_1)^{-1}s$ such that $v_2 \sqsubseteq w_2$.

Then $s' \sqsubseteq a_2a_2$, since $s \sqsubseteq w'$. If $s' = \varepsilon$ or $s' = a_2$, then $s \sqsubseteq v \sqsubseteq w$, and we are done. So assume $s' = a_2a_2$. Let $t = v_0a_1v_1a_2v_2$. Then $t \sqsubseteq w$ and $|t| = |s| \leqslant n$, so t is a subword of both u and v.

▶ **Claim.** $v_1a_2v_2 \sqsubseteq a_1w_1w_2$.

Proof. The claim asserts that a certain suffix of t is a subword of a certain suffix of u. We know that $t \sqsubseteq u$, i.e., $v_0a_1v_1a_2v_2 \sqsubseteq w_0a_1w_1w_2$. Hence if $v_1a_2v_2 \not\sqsubseteq a_1w_1w_2$, then $v_0a_1\alpha \sqsubseteq w_0$ for some nonempty prefix α of $v_1a_2v_2$. But this contradicts the definition of v_0. ◀

Now since $v_1a_2v_2 \sqsubseteq a_1w_1w_2$, we have $v_1a_2 \sqsubseteq a_1w_1$ or $a_2v_2 \sqsubseteq w_2$. Combining this with $v_1 \sqsubseteq w_1$ and $v_2 \sqsubseteq w_2$, we get $v_1a_2a_2v_2 \sqsubseteq a_1w_1a_2w_2$. Finally, this along with $v_0 \sqsubseteq w_0$ gives $s \sqsubseteq w$ as needed.

One-Dimensional Logic over Words*

Emanuel Kieroński

University of Wrocław, Poland
kiero@cs.uni.wroc.pl

Abstract

One-dimensional fragment of first-order logic is obtained by restricting quantification to blocks of existential quantifiers that leave at most one variable free. We investigate one-dimensional fragment over words and over ω-words. We show that it is expressively equivalent to the two-variable fragment of first-order logic. We also show that its satisfiability problem is NExpTime-complete. Further, we show undecidability of some extensions, whose two-variable counterparts remain decidable.

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

Keywords and phrases satisfiability, expressivity, words, fragments of first-order logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.38

1 Introduction

One-dimensional fragment of first-order logic, F_1, is obtained by restricting quantification to blocks of existential quantifiers that leave at most one variable free. It is not difficult to show that over general relational structures the satisfiability problem for F_1 is undecidable [8]. Its *uniform* variant, UF_1, was introduced by Hella and Kuusisto in [8] as a generalization of the two-variable fragment of first-order logic, FO^2, to contexts with relations of arity higher than two. In that paper the decidability and the finite model property for UF_1 without equality was proved. Roughly speaking, the uniformity restrictions allow for Boolean combination of atoms $Rx_1, \ldots, x_m$ and $Sy_1, \ldots, y_n$ of arity greater than one only if the sets of variables $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_n\}$ are equal; Boolean combinations of atoms of arity one can be formed freely. In [9] the finite model property was extended to UF_1 with free (i.e, not necessarily uniform) use of equality. It was also shown that the satisfiability problem is NExpTime-complete. Both results were obtained, to some extent, by a generalisation of the classical techniques used by Grädel, Kolaitis and Vardi in [7] in context of FO^2 (whose satisfiability problem is also NExpTime-complete). A nice survey of the results on UF_1 can be found in a recent paper by Kuusisto [13], which also reveals some connections between UF_1 and description logics.

The uniformity restriction is indeed crucial for the decidability of UF_1. Unfortunately, it also limits the possible scenarios in which this logic can be used. A question is if there are any ways of weakening it without losing decidability. Please note that actually the variant of UF_1 considered in [9] has a non-uniform ingredient, namely the equality predicate. This non-uniformity indeed gives an additional power: it turns out that UF_1 restricted to signatures with only unary and binary relational symbols is expressively equivalent to FO^2 if the uniform use of equality is imposed, while it is more expressive if there is no such restriction, offering in particular some sort of counting. Even a stronger decidable

* Partially supported by the Polish National Science Centre grant DEC-2013/09/B/ST6/01535.

© Emanuel Kieroński;
licensed under Creative Commons License CC-BY

logic was identified in [10]. It is UF_1 with a free use of one equivalence relation, shown to be 2-NExpTime-complete (or even NExpTime-complete when some natural variation is considered). That paper demonstrates however that this decidability result is fragile. E.g., adding a second equivalence relation leads to undecidability. This contrasts with the case of FO^2 which is decidable in the presence of two equivalences and becomes undecidable only after adding the third one [11]. UF_1 becomes also undecidable when extended with a non-uniform use of a single transitive relation. Again, this reveals a difference in comparison with FO^2 whose satisfiability in models with one transitive relation was show decidable by Szwast and Tendera [19].

Instead of extending UF_1 with some non-uniform ingredients we may also try to investigate full F_1 over some classes of structures, in which the meaning of non-unary symbols is fixed. One of the simplest, but very important such class is the class of words. Many formalisms over words have been investigated so far. It is known that the satisfiability problem for full first-order logic is decidable, but with non-elementary complexity, as shown by Stockmeyer [18]. In fact, already the fragment with three variables is non-elementary. On the other hand a reasonable complexity appears when the number of variables is restricted to two. The satisfiability problem for FO^2 over words and ω-words was shown to be NExpTime-complete by Etessami, Vardi and Wilke [6]. In the same paper it is observed that the expressive power of FO^2 over words is equal to the expressive power of unary temporal logic, UTL, i.e., temporal logic with four navigational operators: *next state*, *somewhere in the future*, *previous state*, *somewhere in the past*. FO^2, however, turns out to be exponentially more succinct than UTL. An extension of FO^2 with counting quantifiers, C^2, is shown to be NExpTime-complete over words by Charatonik and Witkowski, [4]. In fact, it is not difficult to observe that over words C^2 has also the same expressive power as plain FO^2. Another interesting extension of FO^2, which significantly increases its expressive power is an extension with the *between* predicate recently studied by Krebs et al. [12].

In this paper we study the expressive power and the complexity of the satisfiability problem of the full one-dimensional fragment of first-order logic over words, $\mathrm{F}_1[<,+1]$. First, we show that its expressive power is the same as the expressive power of FO^2, and thus also of UTL, and, as mentioned, of C^2.

The advantage of F_1 over those other formalisms is that it allows to specify some properties in a more natural and elegant way. If we want to say that a word contains some (especially not fully specified) pattern, consisting of more than two elements, we can just quantify an appropriate number of positions, and say how they should be labelled and related to each other. Expressing the same in FO^2 will usually require some heavy recycling of the two available variables. Let us look at two simple examples. Consider a system whose behaviour we model as a word or an ω-word, in which one or more of n atomic propositions out of $P_1, \ldots, P_n$ can hold in a given point of time. To say that there are m non-overlapping time intervals (sets of consecutive positions of the word) in each of which each of P_i holds at least once, we can use the following $\mathrm{F}_1[<,+1]$ sentence:

$$\exists y_0 y_1 \ldots y_n x_{11} \ldots x_{1n} \ldots x_{m1} \ldots x_{mn} \Big(\bigwedge_{i=1}^{m} \bigwedge_{j=1}^{n} y_{i-1} \leq x_{ij} \wedge x_{ij} < y_i \wedge P_j x \Big).$$

As another example[1] take the property saying that it is possible to choose no more than m positions satisfying together all of P_i:

[1] Pointed out to the author by Jakub Michaliszyn.

$$\exists x_1 \dots x_m (\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} P_i x_j).$$

The reader is asked to check that expressing the above properties in $\mathrm{FO}^2[<,+1]$ is indeed not straightforward and leads to complicated formulas.

In fact, what is worth mentioning here, our translation of $\mathrm{F}_1[<,+1]$ to $\mathrm{FO}^2[<,+1]$ has an exponential blow-up, which seems to be hard to avoid, and which thus suggests that $\mathrm{F}_1[<,+1]$ may be able to express some properties more succinctly than $\mathrm{FO}^2[<,+1]$, and possibly, even C^2.

Further, we turn our attention to the satisfiability problem for F_1, managing to show that, in spite of the exponential blow-up in the translation, $\mathrm{F}_1[<,+1]$ retains the complexity of $\mathrm{FO}^2[<,+1]$, i.e., is NExpTime-complete. While our proof has some similarities to the proof of Etessami, Vardi and Wilke [6] for $\mathrm{FO}^2[<,+1]$, it is technically more involved, due to the combinatorically more complicated nature of the objects involved. Nevertheless, the basic idea in the proof is rather straightforward and is based on an appropriately tuned contraction procedure.

We conclude the paper examining some possible extensions of $\mathrm{F}_1[<,+1]$. Probably, the most significant of them is the extension of $\mathrm{F}_1[<,+1]$ with an equivalence relation, inspired by an analogous extension of $\mathrm{FO}^2[<,+1]$ (FO^2 over *data words*), studied by Bojańczyk et al. [2]. The satisfiability problem for FO^2 over data words, even though very hard, is decidable. We show that $\mathrm{F}_1[<,+1]$ over data words becomes undecidable.

Finally, we suggest some related open problems concerning F_1 over some specific classes of structures.

2 Preliminaries

We assume that the reader is familiar with basic concepts in mathematical logic and computational complexity theory. Throughout this paper we mostly use standard terminology and notation.

By *one-dimensional fragment* of first-order logic, F_1, we mean the relational fragment in which quantification is restricted to blocks of existential quantifiers that leave at most one variable free. Formally, F_1 over relational signature τ and some countably infinite set of variables Var is the smallest set such that:

- $R\bar{x} \in \mathrm{F}_1$ for all $R \in \tau$ and all tuples $\bar{x}$ of variables from Var,
- $x = y \in \mathrm{F}_1$ for all variables $x, y \in Var$,
- F_1 is closed under $\vee$ and $\neg$,
- if φ is an F_1 formula with free variables $x_0, \dots, x_k$ then formulas $\exists x_0, \dots, x_k \varphi$ and $\exists x_1, \dots, x_k \varphi$ belong to F_1.

As usually, we can use standard abbreviations for other Boolean operations, like $\wedge, \rightarrow, \top$, etc., as well as for universal quantification. The length of a formula φ is measured in a natural way, and denoted $\|\varphi\|$. The *width* of a formula is the maximum of the numbers of free variables in its subformulas.

We will be primarily interested in signatures consisting of a possibly infinite set of unary symbols and two binary symbols $+1$ and $<$. The obtained logic is then denoted $\mathrm{F}_1[<,+1]$. Sometimes we will use another binary symbol $\ll$, which is an abbreviation: $x \ll y \equiv x < y \wedge \neg(+1(x,y))$. In Section 5 we will consider also some other binary symbols, whose meaning will be then explained.

We denote structures with Gothic capital letters, possibly with decorations: $\mathfrak{M}, \mathfrak{M}', \mathfrak{M}_1$, etc., and their universes with the corresponding Roman capital letters M, M', M_1, etc. We are interested in structures in which $<$ is interpreted as a linear order and $+1$ as its induced successor relation. Such a structure is called a *finite word*, or just a *word*, if its universe is finite, and *ω-word* if after dropping the interpretation of unary relations it is isomorphic to $(\mathbb{N}, <, +1)$. If $\mathfrak{M}$ is a word and $\mathfrak{M}'$ a word or ω-word we denote by $\mathfrak{M}\mathfrak{M}'$ the word obtained by the concatenation of $\mathfrak{M}$ and $\mathfrak{M}'$; if $\mathfrak{M}'$ consists of just one element a then this concatenation is written as $\mathfrak{M}a$. An ω-word built out of a finite word $\mathfrak{M}_0$ followed by infinitely many copies of a finite word $\mathfrak{M}_1$ is denoted by $\mathfrak{M}_0\mathfrak{M}_1^\omega$. Such an ω-word is called *periodic*. When referring to the elements of a model $\mathfrak{M}$ we will sometimes denote by $a + i$, for $a \in M$ and $i \in \mathbb{Z}$, the element located i positions to the right from a if $i > 0$, $-i$ positions to the left from a if $i < 0$, and the element a if $i = 0$. The *satisfiability problem* over words (ω-words) for a logic $\mathcal{L}$ is to check if for a given sentence $\varphi \in \mathcal{L}$ there exists a word (ω-word) $\mathfrak{M}$ such that $\mathfrak{M} \models \varphi$.

3 Expressivity

It is known that $\mathrm{FO}^2[<, +1]$ is expressively equivalent over words and ω-words to unary temporal logic, UTL, i.e., temporal logic with four navigational operators: *next state*, *somewhere in the future*, *previous state*, *somewhere in the past* [6]. Here we show that $\mathrm{F}_1[<, +1]$ shares their expressivity:

▶ **Theorem 1.** *$\mathrm{F}_1[<, +1]$ and $\mathrm{FO}^2[<, +1]$ are expressively equivalent over words and ω-words.*

Obviously, $\mathrm{FO}^2[<, +1]$ can be seen as a fragment of $\mathrm{F}_1[<, +1]$. Here we present a translation from $\mathrm{F}_1[<, +1]$ to $\mathrm{FO}^2[<, +1]$ which justifies Thm. 1. More specifically, we show that for any $\mathrm{F}_1[<, +1]$ sentence there is an FO^2 sentence satisfied in precisely the same models, and that for any $\mathrm{F}_1[<, +1]$ formula with one free variable there is an FO^2 formula with one free variable such that they are satisfied at the same positions of any model. The crux is to show how to translate formulas starting with a block of quantifiers.

▶ **Lemma 2.** *For any $\mathrm{F}_1[<, +1]$ formula $\psi = \exists y_1 \ldots, y_k \psi_0(y_0, y_1, \ldots, y_k)$ with free variable y_0 there exists an FO^2 formula ψ' with one free variable such that for every word or ω-word $\mathfrak{M}$ and every $a \in M$ we have $\mathfrak{M} \models \psi[a]$ iff $\mathfrak{M} \models \psi'[a]$. Similarly, for any $\mathrm{F}_1[<, +1]$ sentence $\psi = \exists y_1 \ldots, y_k \psi_0(y_1, \ldots, y_k)$ there exists an FO^2 sentence ψ' such that for any word or ω-word $\mathfrak{M}$ we have $\mathfrak{M} \models \psi$ iff $\mathfrak{M} \models \psi'$.*

Proof. We prove this lemma by induction over the quantifier depth of ψ, measured as the maximal nesting depth of blocks of quantifiers rather than of individual quantifiers. We explicitly consider the case of a subformula with a free variable (the case of sentences can be treated similarly). Let us take any

$$\psi = \exists y_1 \ldots, y_k \psi_0(y_0, y_1, \ldots, y_k), \tag{1}$$

convert ψ_0 into disjunctive form and distribute existential quantifiers over disjunctions, obtaining

$$\psi = \bigvee_{i=1}^{l} \exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k), \tag{2}$$

for some $l \in \mathbb{N}$, where each ψ_i is a conjunction of literals, subformulas with one free variable of the form $\exists z_1, \ldots, z_k \psi_0(y_j, z_1, \ldots, z_k)$, subsentences of the form $\exists z_1, \ldots, z_k \psi_0(z_1, \ldots, z_k)$, and negations of such formulas.

Recall that possible atoms are Ay_i for a unary symbol A, $y_i < y_j$, $+1(y_i, y_j)$ and $y_i = y_j$, for some i, j.

An *ordering scheme* over variables $y_0, \ldots, y_k$ is a formula of the form $\eta_0(y_{i_0}, y_{i_1}) \wedge \eta_1(y_{i_1}, y_{i_2}) \wedge \ldots \wedge \eta_{k-1}(y_{i_{k-1}}, y_{i_k})$, where $\eta_i(v, w)$ is one of the following formulas: $v = w$, $+1(v, w)$ or $v \ll w$, and $i_0, i_1, i_2, \ldots, i_k$ is a permutation of $0, 1, \ldots, k$.

Consider now a single disjunct $\exists y_1 \ldots, y_k \psi_i(y_0, y_1, \ldots, y_k)$ of (2) (again assuming that it is not a sentence and has free variable y_0), and replace it by the following disjunction over all possible ordering schemes π over $y_0, \ldots, y_k$:

$$\bigvee_{\pi} \exists y_1 \ldots, y_k(\pi(y_0, \ldots, y_k) \wedge \psi_i^{\pi}(y_0, y_1, \ldots, y_k)), \tag{3}$$

where ψ_i^{π} is obtained from ψ_i by replacing all atoms $y_i < y_j$, $+1(y_i, y_j)$ and $y_i = y_j$, which are not bounded by the quantifiers of the subformulas of ψ_i by $\top$ or $\bot$, according to the information recorded in π. Let us rearrange ψ_i^{π} into $\psi_{i,*}^{\pi} \wedge \bigwedge_{j=0}^{k} \psi_{i,j}^{\pi}(y_j)$, where $\psi_{i,*}^{\pi}$ consists of the conjuncts without free variables and $\psi_{i,j}^{\pi}(y_j)$ consists of the conjuncts with free variable y_j. We now explain how to translate a single disjunct

$$\exists y_1 \ldots, y_k(\pi(y_0, \ldots, y_k) \wedge \psi_{i,*}^{\pi} \wedge \bigwedge_{j=0}^{k} \psi_{i,j}^{\pi}(y_j)) \tag{4}$$

of (3). Let $i_0, i_1, \ldots, i_k$ be the permutation used to generate π, and let s be such that $i_s = 0$. By the inductive assumption we can replace $\psi_{i,*}^{\pi}$ by an equivalent FO^2 sentence $\psi_{i,*}'^{\pi}$, with two variables. We can also replace in each $\psi_{i,j}^{\pi}(y_j)$ any conjunct of the form $\exists z_1, \ldots, z_k \chi(y_j, z_1, \ldots, z_k)$ by an equivalent two-variable conjunct with one free variable. Thus, in turn, $\psi_{i,j}^{\pi}(y_j)$ can be replaced by an equivalent FO^2 formula $\psi_{i,j}'^{\pi}$ with one free variable.

We finally replace (4) by the conjunction of

$$\psi_{i,*}'^{\pi} \wedge \psi_{i,i_s}'^{\pi}(y_0), \tag{5}$$

$$\exists y(\eta_{s-1}(y, y_0) \wedge \psi_{i,i_{s-1}}'^{\pi}(y) \wedge \exists y_0(\eta_{s-2}(y_0, y) \wedge \psi_{i,i_{s-2}}'^{\pi}(y_0) \wedge \ldots)), \tag{6}$$

$$\exists y(\eta_{s+1}(y, y_0) \wedge \psi_{i,i_{s+1}}'^{\pi}(y) \wedge \exists y_0(\eta_{s+2}(y_0, y) \wedge \psi_{i,i_{s+2}}'^{\pi}(y_0) \wedge \ldots)), \tag{7}$$

in which (5) enforces the satisfaction of the proper subsentences, (6) takes care of witnesses smaller than (or equal) to y_0, passing the word from y_0 to the left, and (7) takes care of witnesses greater then (or equal to) y_0, passing the word from y_0 to the right. Of course, in all the above formulas we appropriately rename the variables if necessary, so that only y_0 and y are used. ◀

Having translated formulas starting with blocks of quantifiers, we can easily translate other formulas with at most one free variable, since they are just boolean combinations of the former. This gives a translation from $\mathrm{F}_1[<, +1]$ to $\mathrm{FO}^2[<, +1]$.

Observe that starting from an $\mathrm{F}_1[<, +1]$ formula this translation may produce a formula in $\mathrm{FO}^2[<, +1]$ which is exponentially longer. Essentially, there are two sources of this exponential blow-up. The first is the transformation to disjunctive form, and the second is considering all possible permutations of variables quantified in a single block of quantifiers. The question whether this blow-up is necessary is left open.

Let us mention here that $\mathrm{C}^2[<, +1]$, the two-variable logic with counting quantifiers easily translates to $\mathrm{F}_1[<, +1]$. For example, to express $\exists^{\geq k} y\ \psi(x, y)$ we can just write $\exists y_1, \ldots, y_k(\bigwedge_{1 \leq i < j \leq k} y_i \neq y_j \wedge \bigwedge_{i=1}^{k} \psi(x, y_i))$. Thus all the logics from the following list: UTL, $\mathrm{FO}^2[<, +1]$, $\mathrm{C}^2[<, +1]$, $\mathrm{F}_1[<, +1]$ are expressively equivalent over words and ω-words.

4 Satisfiability

We next turn our attention to satisfiability. We prove that the satisfiability problem for $\mathrm{F}_1[<,+1]$ both over words and ω-words is NExpTime-complete. To this end we start with introducing a convenient normal form, inspired by Scott normal form for FO^2 (a similar normal form is used also in [9]). Then we develop a contraction method involving a careful analysis of certain similarities between elements in a model, and explain how to use it in order to obtain small model properties for $\mathrm{F}_1[<,+1]$ both over words and ω-words. Then the complexity result will easily follow.

4.1 Normal form

We adapt here the well known Scott normal form for FO^2 [17] to our purposes. We say that an $\mathrm{F}_1[<,+1]$ formula φ is in *normal form* if φ has the following shape:

$$\bigwedge_{1\leq i\leq m_\exists} \forall y_0 \exists y_1 \dots y_{k_i} \varphi_i^\exists \wedge \bigwedge_{1\leq i\leq m_\forall} \forall x_1 \dots x_{l_i} \varphi_i^\forall, \tag{8}$$

where $\varphi_i^\exists = \varphi_i^\exists(y_0, y_1, \dots, y_{k_i})$ and $\varphi_i^\forall = \varphi_i^\forall(x_1, \dots, x_{l_i})$ are quantifier-free. Please note that the width of φ is the maximum of the set $\{k_i+1\}_{1\leq i\leq m_\exists} \cup \{l_j\}_{1\leq j\leq m_\forall}$. The following fact can be proved in a standard fashion, see, e.g., [5] for a more detailed exposition of the technique.

▶ **Lemma 3.** *For every $\mathrm{F}_1[<,+1]$ formula φ, one can compute in polynomial time an $\mathrm{F}_1[<,+1]$ formula φ' in normal form (over the signature extended by some fresh unary symbols) such that: (i) any model of φ can be expanded to a model of φ' by appropriately interpreting new unary symbols; (ii) any model of φ' restricted to the signature of φ is a model of φ.*

Proof (Sketch). We successively replace innermost subformulas ψ of φ of the form $\exists y_1, \dots, y_k$ $\varphi(y_0, y_1, \dots, y_k)$ by atoms $P_\psi(y_0)$, where P_ψ is a fresh unary symbol, and axiomatize P_ψ using two normal form conjuncts: $\forall y_0 \exists y_1, \dots, y_k (P_\psi(y_0) \to \varphi(y_0, y_1, \dots, y_k))$ and $\forall y_0, y_1, \dots, y_k$ $(\neg\varphi(y_0, y_1, \dots, y_k) \vee P_\psi(y_0))$. ◀

The above lemma allows us, when dealing with satisfiability or when analysing the size and shape of models, to restrict attention to normal form formulas.

4.2 Contraction

Let τ be a finite unary signature. A 1-*type* over τ is a subset of τ. We say that an element $a \in M$ *realizes* a 1-type α in a word or ω-word $\mathfrak{M}$, and write $\mathfrak{M} \models \alpha[a]$, if for each $A \in \tau$ we have $\mathfrak{M} \models A[a]$ iff $A \in \alpha$. The 1-type realized by a in $\mathfrak{M}$ is denoted by $\mathrm{type}^{\mathfrak{M}}(a)$.

Let us introduce some new, more sophisticated concepts which will turn out to be helpful in our constructions.

▶ **Definition 4.**

- An *ordered k-type* is a tuple of the form $(\alpha_1, \eta_1, \alpha_2, \dots, \alpha_{k-1}, \eta_{k-1}, \alpha_k)$, where α_i is an atomic 1-type $(1 \leq i \leq k)$ and $\eta_i(x,y)$ is either $+1(x,y)$ or $x \ll y$ $(1 \leq i \leq k-1)$.
- For a given word or ω-word $\mathfrak{M}$ we say that a tuple of its distinct elements $a_1, \dots, a_k \in M$ *realizes* an ordered k-type $(\alpha_1, \eta_1, \alpha_2, \dots, \alpha_{k-1}, \eta_{k-1}, \alpha_k)$ if $\mathfrak{M} \models \alpha_i[a_i]$ for $1 \leq i \leq k$ and $\mathfrak{M} \models \eta_i[a_i, a_{i+1}]$ for $1 \leq i < k$.

Thus, an ordered k-type stores some basic information about k distinct elements in a model and about their relative location. When k is clear from the context we sometimes talk just about the *ordered type* of a tuple. The ordered type realized by a tuple $a_1, \ldots, a_k$ in $\mathfrak{M}$ is denoted as $\mathrm{ordtype}^{\mathfrak{M}}(a_1, \ldots, a_k)$. We are going to abstract the information about a single element of a model using the following notion.

▶ **Definition 5.** For a given word or ω-word $\mathfrak{M}$, $a \in M$, and a natural number $n > 0$, we say that the *n-profile* of a is the tuple $(\alpha_{-n}, \ldots, \alpha_{-1}, \alpha_0, \alpha_1, \ldots, \alpha_n, L_{-1}, \ldots, L_{-n}, R_1, \ldots, R_n)$, where

- α_i is the 1-type of element $a+i$ of $\mathfrak{M}$ if $a+i$ is defined, or $\alpha_i = \emptyset$ otherwise,
- L_{-i} is the set of the ordered k-types ($1 \leq k \leq n$) realized in the prefix of $\mathfrak{M}$ ending at $a-i-1$,
- R_i is the set of the ordered k-types ($1 \leq k \leq n$) realized in the suffix of $\mathfrak{M}$ starting at $a+i+1$.

The n-profile of an element $a \in M$ is denoted as $\mathrm{prof}_n^{\mathfrak{M}}(a)$. We sometimes say that an element *realizes* its profile.

Let us observe that the number of possible n-profiles realized in a model is not very large.

▶ **Lemma 6.** *If $\mathfrak{M}$ is a word or ω-word over a signature τ then the number of different n-profiles of elements of $\mathfrak{M}$ is bounded exponentially in $|\tau|$ and in n.*

Proof. The number of atomic 1-types realized in $\mathfrak{M}$ is at most $2^{|\tau|}$. The number of ordered k-types is bounded by $2^{k|\tau|} \cdot 2^{(k-1)} \leq 2^{k(|\tau|+1)}$. Thus the size of each of the sets L_i and R_i in an n-profile is bounded exponentially by $n \cdot 2^{n(|\tau|+1)}$. Moreover, these sets behave monotonically in $\mathfrak{M}$: If $(\alpha_{-n}, \ldots, \alpha_{-1}, \alpha_0, \alpha_1, \ldots, \alpha_n, L_{-1}, \ldots, L_{-n}, R_1, \ldots, R_n)$ and $(\alpha'_{-n}, \ldots, \alpha'_{-1}, \alpha'_0, \alpha'_1, \ldots, \alpha'_n, L'_{-1}, \ldots, L'_{-n}, R'_1, \ldots, R'_n)$ are n-profiles of elements $a, a' \in M$, respectively, such that $\mathfrak{M} \models a < a'$, then $L_i \subseteq L'_i$ and $R_i \supseteq R'_i$ ($-n \leq i \leq n$). Hence, when moving from the first position of $\mathfrak{M}$ to the right, each of the sets L_i and R_i may change at most exponentially many times. Since the number of n-profiles with all L_i and R_i fixed is bounded by $2^{(2n+1)|\tau|}$ the claim follows. ◀

We are ready to prove the crucial contraction lemma. Namely, we observe that removing a fragment of a word between two realizations of the same profile, does not change the profiles of the surviving elements. As we will see later such surgery also does not affect the satisfaction of certain normal form formulas.

▶ **Lemma 7.** *Let $\mathfrak{M} = \mathfrak{M}_1 c \mathfrak{M}_2 d \mathfrak{M}_3$ be a word or ω-word and $n > 0$ a natural number. Assume that $\mathrm{prof}_n^{\mathfrak{M}}(c) = \mathrm{prof}_n^{\mathfrak{M}}(d)$ and $\mathfrak{M}' = \mathfrak{M}_1 c \mathfrak{M}_3$. Then for each $a' \in M'$ we have $\mathrm{prof}_n^{\mathfrak{M}'}(a') = \mathrm{prof}_n^{\mathfrak{M}}(a')$.*

Proof. Consider the case when $a' \in M_1 \cup \{c\}$. Note that the prefix of $\mathfrak{M}$ ending in a' is equal to the prefix of $\mathfrak{M}'$ ending in a'. It follows that for all $-n \leq i < 0$ we have $\mathrm{prof}_n^{\mathfrak{M}'}(a').\alpha_i = \mathrm{prof}_n^{\mathfrak{M}}(a').\alpha_i$ and $\mathrm{prof}_n^{\mathfrak{M}'}(a').L_i = \mathrm{prof}_n^{\mathfrak{M}}(a').L_i$, since these components refer only to positions located in this initial prefix.

Take such $l \geq 0$ that in $\mathfrak{M}'$ we have $c = a' + l$. For $0 \leq i \leq l$, it is the case that $a'+i$ in $\mathfrak{M}$ equals $a'+i$ in $\mathfrak{M}'$ and thus $\mathrm{prof}_n^{\mathfrak{M}'}(a').\alpha_i = \mathrm{prof}_n^{\mathfrak{M}}(a').\alpha_i$; if $l < i \leq n$ then $a'+i = c+(i-l)$ in $\mathfrak{M}'$ and it equals $d+(i-l)$ in $\mathfrak{M}$, and since $\mathrm{prof}_n^{\mathfrak{M}'}(c).\alpha_{i-l} = \mathrm{prof}_n^{\mathfrak{M}}(d).\alpha_{i-l}$ it also gives $\mathrm{prof}_n^{\mathfrak{M}'}(a').\alpha_i = \mathrm{prof}_n^{\mathfrak{M}}(a').\alpha_i$.

It remains to see that $\mathrm{prof}_n^{\mathfrak{M}'}(a').R_i = \mathrm{prof}_n^{\mathfrak{M}}(a').R_i$ for $0 < i \leq n$. To show that $\mathrm{prof}_n^{\mathfrak{M}'}(a').R_i \subseteq \mathrm{prof}_n^{\mathfrak{M}}(a').R_i$ take any k-ordered type γ belonging to $\mathrm{prof}_n^{\mathfrak{M}'}(a').R_i$ and let $a'_1, \ldots, a'_k$ be a realization of γ in $\mathfrak{M}'$ such that $\mathfrak{M}' \models a'+i < a'_1 < \ldots < a'_k$. See Fig. 1. Let us divide the sequence $a'_1, \ldots, a'_k$ into three (possibly empty) fragments:

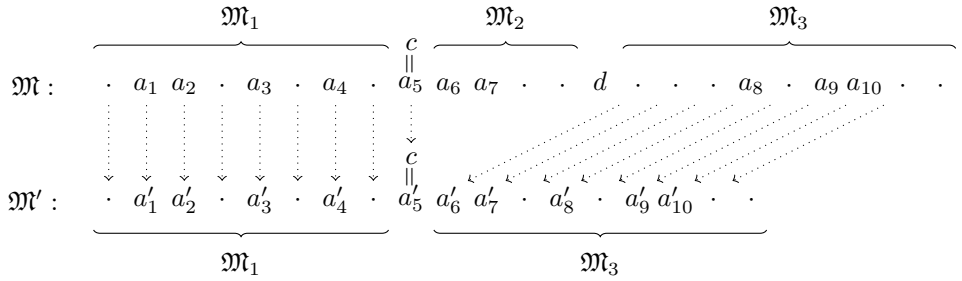


Figure 1 Contraction of $\mathfrak{M}$ into $\mathfrak{M}'$. The tuple $a_1, \ldots, a_{10}$ has the same ordered type as $a'_1, \ldots, a'_{10}$.

- $a'_1, \ldots, a'_s$ containing all elements a'_j such that $\mathfrak{M}' \models a'_j \leq c$,
- $a'_{s+1}, \ldots, a'_{s+t}$ which is empty if $a'_s \neq c$; and is the maximal fragment of consecutive elements located just to the right from c, i.e., such that in $\mathfrak{M}'$ we have $a'_{s+i} = c + i$ for $1 \leq i \leq t$, and $a'_{s+t+1} \neq c + t + 1$,
- $a'_{t+1}, \ldots, a'_k$ being the remaining fragment.

In the example from Fig. 1 we have $s = 5$, and $t = 2$. We show that there is a realization $a_1, \ldots, a_k$ of γ in $\mathfrak{M}$ such that $a' + i < a_1 < \ldots < a_k$. For $j \leq s$ it suffices to take $a_j := a'_j$. For $s < j \leq s+t$ we take $a_j := c+(j-s)$ (as computed in $\mathfrak{M}$). Finally, for $s+t < j \leq k$, we again take $a_j := a'_j$. It is readily verified that $\mathrm{ordtype}^{\mathfrak{M}}(a_1, \ldots, a_k) = \mathrm{ordtype}^{\mathfrak{M}'}(a'_1, \ldots, a'_k) = \gamma$, and thus $\gamma \in \mathrm{prof}^{\mathfrak{M}}_n(a').R_i$.

To show that $\mathrm{prof}^{\mathfrak{M}}_n(a').R_i \subseteq \mathrm{prof}^{\mathfrak{M}'}_n(a').R_i$ we take any k-ordered type γ belonging to $\mathrm{prof}^{\mathfrak{M}}_n(a').R_i$ and let $a_1, \ldots, a_k$ be a realization of γ in $\mathfrak{M}$ such that $\mathfrak{M} \models a' + i < a_1 < \ldots < a_k$. We again split the sequence $a_1, \ldots, a_k$ into three (possibly empty) fragments:

- $a_1, \ldots, a_s$ containing all elements a_j such that $\mathfrak{M} \models a_j \leq c$,
- $a_{s+1}, \ldots, a_{s+t}$ which is empty if $a_s \neq c$; and is the maximal fragment of consecutive elements located just to the right from c, i.e., such that in $\mathfrak{M}$ we have $a_{s+i} = c + i$ for $1 \leq i \leq t$, and $a_{s+t+1} \neq c + t + 1$,
- $a_{t+1}, \ldots, a_k$ being the remaining fragment.

Note that this time the second fragment belongs to $\mathfrak{M}_2 d \mathfrak{M}_3$. We show that there is a realization $a'_1, \ldots, a'_k$ of γ in $\mathfrak{M}'$ such that $\mathfrak{M}' \models a' + i < a'_1 < \ldots < a'_k$. For $j \leq s$ it suffices to take $a'_j := a_j$. For $s < j \leq s + t$ we take $a'_j := c + (j - s)$ (note that $c + (j - s)$ is computed in $\mathfrak{M}'$ so it belongs to M_3). Finally, let $\gamma' = \mathrm{ordtype}^{\mathfrak{M}}(a_{s+t+1}, \ldots, a_k)$. Observe that $\gamma' \in \mathrm{prof}^{\mathfrak{M}}_n(c).R_{t+1}$. Thus $\gamma' \in \mathrm{prof}^{\mathfrak{M}}_n(d).R_{t+1}$, and this means that there is a realization of γ in $\mathfrak{M}_3$, to the right from $d + t + 1$. In $\mathfrak{M}'$ the same realization is located to the right from $c + t + 1$ and we can take its elements as $a'_{s+t+1}, \ldots, a'_k$. Again, t is readily verified that $\mathrm{ordtype}^{\mathfrak{M}}(a_1, \ldots, a_k) = \mathrm{ordtype}^{\mathfrak{M}'}(a'_1, \ldots, a'_k) = \gamma$, and thus $\gamma \in \mathrm{prof}^{\mathfrak{M}'}_n(a').R_i$.

The case when $a' \in M_3$ can be treated symmetrically: this time we get equality of the R_i components of profiles for free and to show equality of the L_i components we use the equality of the L_i components of the profiles of c and d. ◀

Let us now observe how the notion of an n-profile is closely related to the satisfaction of normal form $\mathrm{F}_1[<, +1]$ formulas.

▶ **Lemma 8.** *Let φ be a normal form $\mathrm{F}_1[<, +1]$ formula of width n and let $\mathfrak{M}$ be a word or ω-word such that $\mathfrak{M} \models \varphi$. Let $\mathfrak{M}'$ be a word or ω-word such that for each $a' \in M'$ there is $a \in M$ such that $\mathrm{prof}^{\mathfrak{M}'}_n(a') = \mathrm{prof}^{\mathfrak{M}}_n(a)$. Then $\mathfrak{M}' \models \varphi$.*

Proof. This observation follows from the definition of the notion of a profile and the definition of the shape of normal form formulas. Below we present some details.

Existential conjuncts. Let us first check that all elements of $\mathfrak{M}'$ have appropriate witnesses for the existential conjuncts. Consider any conjunct $\forall y_0 \exists y_1 \dots y_{k_i} \varphi_i^\exists$ of φ and any element $a'_0 \in M'$. Let $a_0 \in M$ be such that $\mathrm{prof}_n^{\mathfrak{M}}(a_0) = \mathrm{prof}_n^{\mathfrak{M}'}(a'_0)$. Let $a_1, \dots, a_{k_i} \in M$ be a tuple of (not necessarily distinct) elements of M forming a witness tuple for a and $\varphi_i^\exists$, i.e., such that $\mathfrak{M} \models \varphi_i^\exists[a_0, a_1, \dots, a_{k_i}]$. Let $b_0, \dots, b_{k'_i}$ be the sequence of distinct elements of M such that $\mathfrak{M} \models b_0 < b_1 < \ldots < b_{k'_i}$ and $\{b_0, \dots, b_{k'_i}\} = \{a_0, a_1, \dots, a_{k_i}\}$. (Note that k'_i may be smaller than k_i because of potential equalities among a'_is.) Assume that $a_0 = b_u$, and that $b_s, \dots, b_u, \dots, b_t$ is the maximal subsequence of $b_0, \dots, b_{k'_i}$ consisting of consecutive elements of $\mathfrak{M}$, containing $b_u = a_0$. Our aim now is to demonstrate that there exists a sequence $b'_0, \dots, b'_{k'_i}$ in $\mathfrak{M}'$ such that $\mathrm{ordtype}^{\mathfrak{M}}(b_0, \dots, b_{k'_i}) = \mathrm{ordtype}^{\mathfrak{M}'}(b'_0, \dots, b'_{k'_i})$ and $b'_u = a'_0$. Such a sequence can be defined in the following way:

- Let $\gamma = \mathrm{ordtype}^{\mathfrak{M}}(b_0, \dots, b_{s-1})$. Observe that $\gamma \in \mathrm{prof}_n^{\mathfrak{M}}(a_0).L_{-(u-s+1)}$ and thus also $\gamma \in \mathrm{prof}_n^{\mathfrak{M}'}(a'_0).L_{-(u-s+1)}$. This guarantees that there is a realization of γ in $\mathfrak{M}'$ to the left from $a'_0 - (u - s + 1)$. We take the elements of this realization as $b'_0, \dots, b'_{s-1}$.
- for $s \leq j \leq t$ we take $b'_s = a'_0 + (j - u)$.
- Let $\gamma' = \mathrm{ordtype}^{\mathfrak{M}}(b_{t+1}, \dots, b_{k'_i})$. Observe that $\gamma \in \mathrm{prof}_n^{\mathfrak{M}}(a_0).R_{t-u+1}$ and thus also $\gamma \in \mathrm{prof}_n^{\mathfrak{M}'}(a'_0).R_{t-u+1}$. This guarantees that there is a realization of γ in $\mathfrak{M}'$ to the right from $a' + (t - s + 1)$. We take the elements of this realization as $b'_{t+1}, \dots, b'_{k'_i}$.

It is readily verified that $\mathrm{ordtype}^{\mathfrak{M}'}(b'_0, \dots, b'_{k'_i})$ is as desired. For $1 \leq j \leq k_i$ we take $a'_j := b'_k$ for such that k that $a_j = b_k$. It follows that $\mathfrak{M}' \models \varphi_i^\exists[a'_0, a'_1, \dots, a'_{k_i}]$.

Universal conjuncts. Consider now a conjunct $\forall x_1 \dots x_{l_i} \varphi_i^\forall(x_1, \dots, x_{l_i})$ of φ. Let $a'_1, \dots, a'_{l_i}$ be any sequence of (not necessarily distinct) elements of M'. We want to see that $\mathfrak{M}' \models \varphi_i^\forall[a'_1, \dots, a'_{l_i}]$. Let $b'_1, \dots, b'_{l'_i}$ be the sequence of elements such that $\mathfrak{M}' \models b'_1 < \ldots < b'_{l'_i}$ and $\{b'_1, \dots, b'_{l'_i}\} = \{a'_1, \dots, a'_{l_i}\}$. It is sufficient to show that there is a sequence $b_1, \dots, b_{l'_i}$ of elements of $\mathfrak{M}$ such that $\mathrm{ordtype}^{\mathfrak{M}}(b_1, \dots, b_{l'_i}) = \mathrm{ordtype}^{\mathfrak{M}'}(b'_1, \dots, b'_{l'_i})$. Assume that $b'_1, \dots, b'_s$ it the maximal prefix of $b'_1, \dots, b'_{l'_i}$ consisting of consecutive elements of $\mathfrak{M}'$. By assumption there is $b_1 \in M$ such that $\mathrm{prof}_n^{\mathfrak{M}}(b_1) = \mathrm{prof}_n^{\mathfrak{M}'}(b'_1)$. We now take $b_j := b_1 + (j-1)$ for $1 \leq j \leq s$. Let $\gamma = \mathrm{ordtype}^{\mathfrak{M}'}(b'_{s+1}, \dots, b'_{l'_i})$. Observe that $\gamma \in \mathrm{prof}_n^{\mathfrak{M}'}(b'_1).R_s$. Thus also $\gamma \in \mathrm{prof}_n^{\mathfrak{M}}(b_1).R_s$, which implies that there is a realization of γ in $\mathfrak{M}$ to the right from $b_1 + s$. The elements of this realization are taken respectively as $b_{s+1}, \dots, b_{l_i}$. It is readily verified that $\mathrm{ordtype}^{\mathfrak{M}}(b_1, \dots, b_{l'_i}) = \mathrm{ordtype}^{\mathfrak{M}'}(b'_1, \dots, b'_{l'_i})$ as desired. ◀

4.3 Surgery on ω-words

In this subsection we work over ω-words. Namely we show how to transform a given ω-word into a periodic one without introducing any new profiles.

▶ **Lemma 9.** *Let $\mathfrak{M}$ be an ω-word and $n > 0$ a natural number. Let $\mathfrak{M}_0$ be the shortest prefix of $\mathfrak{M}$ such that it contains all the elements having the n-profiles which are realized finitely many times in $\mathfrak{M}$. Let a be the first element not belonging to M_0, and π its n-profile. Let $\mathfrak{M}_1$ be the shortest fragment of $\mathfrak{M}$ such that*

- *it starts at a,*
- *contains the realizations of all n-profiles realized in $\mathfrak{M}$ infinitely many times,*

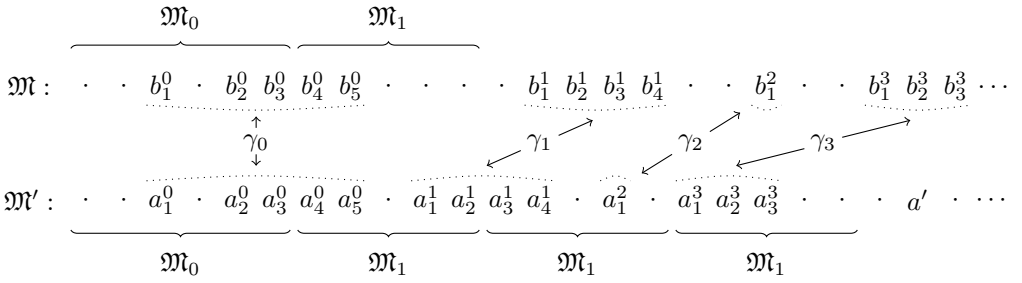


Figure 2 Building a periodic model $\mathfrak{M}'$ from $\mathfrak{M}$.

- *ends at an element whose successor has n-profile π,*
- *and its length is greater than n.*

Consider the ω-word $\mathfrak{M}' = \mathfrak{M}_0\mathfrak{M}_1^\omega$. For each $a' \in M'$, if $a' \in M_0$ then $\mathrm{prof}_n^{\mathfrak{M}'}(a') = \mathrm{prof}_n^{\mathfrak{M}}(a')$*, and if a' belongs to a copy of $\mathfrak{M}_1$ then* $\mathrm{prof}_n^{\mathfrak{M}'}(a') = \mathrm{prof}_n^{\mathfrak{M}}(a)$ *for $a \in M_1$ being the element whose a' is a copy of.*

Proof. We consider explicitly the case when a' is a copy of an element $a \in M_1$. (The case of $a' \in M_0$ is similar and simpler.) It should be clear that for $-n \leq i \leq n$ we have $\mathrm{prof}_n^{\mathfrak{M}'}(a') = \mathrm{prof}_n^{\mathfrak{M}}(a)$. Since the prefix of $\mathfrak{M}$ ending at the predecessor of a is also a prefix of $\mathfrak{M}'$, and a' is located to the right from it in $\mathfrak{M}'$, it follows that for all $-n \leq i < 0$ we have $\mathrm{prof}_n^{\mathfrak{M}}(a).L_i \subseteq \mathrm{prof}_n^{\mathfrak{M}'}(a').L_i$. Let us show the opposite containment. Take any $\gamma \in \mathrm{prof}_n^{\mathfrak{M}'}(a').L_i$ and its realization $\bar{a}'_\pi$ located to the left from $a' + i$. Let us write the elements of $\bar{a}'_\pi$, in the increasing order, as $a_1^0, \ldots, a_u^0, \ldots, a_{s_0}^0, a_1^1, \ldots, a_{s_1}^1, \ldots, a_1^k, \ldots, a_{s_k}^k$, where (cf. Fig. 2):

- $a_1^0, \ldots, a_u^0$ are all members of $\bar{a}'_\pi$ from M_0; in the example from Fig. 2 we have $u = 3$,
- if a_u^0 is not the rightmost element of $\mathfrak{M}_0$ then $u = s_0$; otherwise s_0 is chosen so that $\mathfrak{M}' \models \bigwedge_{u \leq j < s_0} {+1}(a_j^0, a_{j+1}^0)$ and $\mathfrak{M}' \models a_{s_0}^0 \ll a_1^1$; note that since $|M_1| > n$ it follows that all $a_1^0, \ldots, a_u^0, \ldots, a_{s_0}^0$ belong to $M_0 \cup M_1$; in the example from Fig. 2 we have $s_0 = 5$,
- for $i > 0$ the fragments $a_1^i, \ldots, a_{s_i}^i$ are maximal fragments of $\bar{a}_\pi$ such that $\mathfrak{M}' \models \bigwedge_{1 \leq j < s_i} {+1}(a_j^i, a_{j+1}^i)$.

For $i \geq 0$ let $\gamma_i := \mathrm{ordtype}^{\mathfrak{M}'}(a_1^i, \ldots, a_{s_i}^i)$. We want to find a realization of γ in $\mathfrak{M}$. Let us set $b_1^0, \ldots, b_u^0, \ldots, b_{s_0}^0$ to be a realization of γ_0 from $M_0 \cup M_1$ consisting just of elements $a_1^0, \ldots, a_u^0, \ldots, a_{s_0}^0$. Observe that for $i > 0$ each of γ_i is realized in $\mathfrak{M}$ infinitely many times. To see this let us denote by $\alpha_1, \ldots, \alpha_{s_i}$ the 1-types of $a_1^i, \ldots, a_{s_i}^i$, respectively, in $\mathfrak{M}'$. Recall that a_1^i is a copy of an element b from M_1 whose n-profile is realized in $\mathfrak{M}$ infinitely many times. Due to our construction the elements $b + j$ from $\mathfrak{M}$, for $1 \leq j \leq s_i$ have 1-types $\alpha_1, \ldots, a_{s_i}$, respectively. Thus there are infinitely many tuples of consecutive elements of 1-types $\alpha_1, \ldots, a_{s_i}$ in $\mathfrak{M}$; any such tuple obviously realizes γ_i. Now we can compose a realization of γ in $\mathfrak{M}$ starting from the chosen realization of γ_0 and iteratively finding a realization of γ_i separated by at least one element from the realization of γ_{i-1}. Moreover, since $\mathrm{prof}_n^{\mathfrak{M}}(a).L_i$ is realized infinitely many times in $\mathfrak{M}$ we can find its realization starting at least n positions to the right from the last element of the chosen realization of γ. This implies that $\gamma \in \mathrm{prof}_n^{\mathfrak{M}}(a).L_i$.

The equality $\mathrm{prof}_n^{\mathfrak{M}}(a).R_i = \mathrm{prof}_n^{\mathfrak{M}'}(a').R_i$ can be proved in a similar fashion. ◀

4.4 Complexity

Lemmas 7, 8 immediately lead to the following small model property.

▶ **Lemma 10.** *Every normal form* $\mathrm{F}_1[<,+1]$ *formula* φ *satisfiable over a finite word has a model of size bounded exponentially in* $\|\varphi\|$.

Proof. It suffices to take any finite model of φ and perform the contraction procedure from Lemma 7 as long as possible, i.e., as long as the model contains a pair of elements with the same n-profile (with n – the width of φ). By Lemma 6 the number of elements in the resulting model is bounded exponentially in $\|\varphi\|$. By Lemmas 7, 8 it indeed satisfies φ. ◀

Similarly, using Lemmas 9, 8 we get:

▶ **Lemma 11.** *Every normal form* $\mathrm{F}_1[<,+1]$ *formula* φ *satisfiable over an* ω*-word has a model* $\mathfrak{M}$ *of the form* $\mathfrak{M} = \mathfrak{M}_1\mathfrak{M}_2^\omega$ *where both* $|M_1|$ *and* $|M_2|$ *are bounded exponentially in* $\|\varphi\|$.

Proof. We start from an arbitrary model and build a periodic model as guaranteed by Lemma 9. Then we shorten its initial and periodic part using Lemma 7, obtaining a model as required. ◀

Finally, we are ready to state our complexity result.

▶ **Theorem 12.** *The satisfiability problems for* $\mathrm{F}_1[<,+1]$ *over words (*ω*-words) is* NExpTime*-complete.*

Proof. For a given $\mathrm{F}_1[<,+1]$ formula φ convert it into its normal form φ'. Then guess a finite model of φ' of size bounded exponentially as guaranteed by Lemma 10 (exponentially bounded initial and periodic parts of a regular ω-model as guaranteed by Lemma 11) and verify that it is indeed a model of φ' (they generate a model of φ'). ◀

5 Undecidable extensions

5.1 Data words

A *data word* (ω*-data word*) is a word (ω-word) with an additional binary relation $\sim$ which is required to be interpreted as an equivalence relation, and which is intended to model data equality tests. Data words are motivated by their connections to XML. FO^2 over data words becomes at least as hard as reachability in Petri nets [2]. Nevertheless, the satisfiability problem remains decidable. We show that $\mathrm{F}_1[<,+1]$ over data words becomes undecidable. Undecidability can be even shown in the absence of $<$.

▶ **Theorem 13.** *The satisfiability problem for* $\mathrm{F}_1[+1,\sim]$ *over finite data words and over* ω*-data-words is undecidable.*

Proof. We employ the standard apparatus of tiling systems. A *tiling system* is a quadruple $\mathcal{T} = \langle C, c_0, \mathit{Hor}, \mathit{Ver}\rangle$, where C is a non-empty, finite set of *colours*, c_0 is an element of C, and Hor, Ver are binary relations on C called the *horizontal* and *vertical* constraints, respectively. We say that $\mathcal{T}$ *tiles* the $m \times n$ grid if there is a function function $f : \{0,1,\ldots,m-1\} \times \{0,1,\ldots,n-1\} \to C$ such that $f(0,0) = c_0$, for all $0 \leq i < m-1$, $0 \leq j \leq n-1$ we have $\langle f(i,j), f(i+1,j)\rangle$ is in Hor, and for all $0 \leq i < m$, $0 \leq j < n-1$ we have $\langle f(i,j), f(i,j+1)\rangle$ is in Ver. It is well know that the problem of checking if for a given tiling system $\mathcal{T}$ there

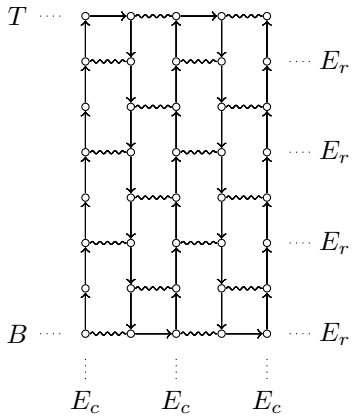


Figure 3 The grid-like structure used to show undecidability of $\mathrm{F}_1[+1, \sim]$.

are m, n such that $\mathcal{T}$ tiles the $m \times n$ grid is undecidable. The problem remains undecidable if we require m to be even and n odd.

To show undecidability of the satisfiability problem for $\mathrm{F}_1[+1, \sim]$ over finite words we construct a formula $\Phi_{\mathcal{T}}$ which is satisfied in a finite word iff $\mathcal{T}$ tiles the $m \times n$ grid for some even m and odd n. We begin the construction of $\Phi_{\mathcal{T}}$ with enforcing that its model is a finite grid-like structure, in which the relation $+1$ forms a snake-like path from its lower-left corner to the upper-right corner, and the equivalence relation connects some elements from neighbouring columns. See Fig. 3. As mentioned, we assume that the number of columns is odd and the number of rows is even. We employ the following unary predicates: B, T, E_c, E_r, whose intended purpose is to mark elements in the bottom row, top row, even columns, and even rows, respectively.

The first two formulas say that the lower left and upper right corners of the grid exist:

$$\exists x(Bx \wedge \neg Tx \wedge E_c x \wedge E_r x \wedge \neg \exists y(+1(y, x))) \tag{9}$$

$$\exists x(Tx \wedge \neg Bx \wedge E_c x \wedge \neg E_r x \wedge \neg \exists y(+1(x, y))) \tag{10}$$

Next we take care of $+1$ relation, ensuring that it respects the intended meaning of the unary predicates:

$$\begin{aligned} \forall xy \quad & (+1(x, y) \to \\ & (E_c x \wedge E_c y \to (\neg By \wedge \neg Tx \wedge (E_r x \leftrightarrow \neg E_r y)) \wedge \\ & (E_c x \wedge \neg E_c y \to (Tx \wedge Ty \wedge \neg Bx \wedge \neg By \wedge \neg E_r x \wedge \neg E_r y)) \wedge \\ & (\neg E_c x \wedge E_c y \to (Bx \wedge By \wedge \neg Tx \wedge \neg Ty \wedge E_r x \wedge E_r y)) \wedge \\ & (\neg E_c x \wedge \neg E_c y \to (\neg Bx \wedge \neg Ty \wedge (E_r \leftrightarrow \neg E_r y)))) \end{aligned} \tag{11}$$

Further, we enforce the appropriate $\sim$-connections. (We abbreviate a formula guaranteeing that $x_1, \ldots, x_k$ agree on E_c-predicate by by $SameColumn(x_1, \ldots, x_k)$.)

$$\forall xyzt(+1(x, y) \wedge +1(y, z) \wedge +1(z, t) \wedge Ty \wedge Tz \to x \sim t) \tag{12}$$

$$\forall xyzt(+1(x, y) \wedge +1(y, z) \wedge +1(z, t) \wedge By \wedge Bz \to x \sim t) \tag{13}$$

$$\begin{aligned} \forall xyztuw(&SameColumn(x, y, z) \wedge SameColumn(t, u, w) \wedge \\ &+1(x, y) \wedge +1(y, z) \wedge z \sim t \wedge +1(t, u) \wedge +1(u, w) \to x \sim w \end{aligned} \tag{14}$$

And finally, we say that T and B are appropriately propagated.

$$\forall xy(x \sim y \to (Tx \leftrightarrow Ty) \wedge (Bx \leftrightarrow By)) \tag{15}$$

$$\forall xyzt(SameColumn(x,y) \wedge SameColumn(z,t) \wedge +1(x,y) \wedge y \sim z \wedge +1(z,t) \to (Tx \leftrightarrow Tt) \wedge (Bx \leftrightarrow Bt)) \tag{16}$$

Formulas (9)-(16) ensure that all the vertical segments of the snake-like path are of the same length and thus that any model indeed looks as in Fig. 3. It remains to encode the tiling problem. We use a unary predicate P_c for each $c \in C$. We say that each node of the grid is coloured by precisely one colour from C and that $(0,0)$ is coloured by c_0:

$$\forall x(\bigvee_{c \in C} P_c(x) \wedge \bigwedge_{c \neq d} \neg(P_c(x) \wedge P_d(x))), \tag{17}$$

$$\forall x((\neg \exists y + 1(y,x)) \to P_{c_0}(x)). \tag{18}$$

Let us abbreviate by $\Theta_H(x,y)$ the formula $\bigwedge_{\langle c,d \rangle \notin Hor}(\neg P_c(x) \wedge \neg P_d(y))$ stating that x, y respect the horizontal constraints of $\mathcal{T}$ and by $\Theta_V(x,y)$ the analogous formula for vertical constraints. We take care of vertical adjacencies:

$$\forall xy(E_c(x) \wedge E_c(y) \wedge +1(x,y) \vee \neg E_c(x) \wedge \neg E_c(y) \wedge +1(y,x) \to \Theta_V(x,y)), \tag{19}$$

and of horizontal adjacencies:

$$\forall xyzt(+1(x,y) \wedge +1(y,z) \wedge +1(z,t) \wedge Ty \wedge Tz \to \Theta_H(y,z)), \tag{20}$$

$$\forall xyzt(+1(x,y) \wedge +1(y,z) \wedge +1(z,t) \wedge By \wedge Bz \to \Theta_H(y,z))), \tag{21}$$

$$\forall xyztuw(SameColumn(x,y,z) \wedge SameColumn(t,u,w) \wedge z \sim t \wedge +1(x,y) \wedge +1(y,z) \wedge +1(t,u) \wedge +1(u,w) \to \Theta_H(x,w) \wedge \Theta_H(y,u) \wedge \Theta_H(z,t)). \tag{22}$$

Let $\Phi_{\mathcal{T}}$ be the conjunction of (9)-(22). From any model of $\Phi_{\mathcal{T}}$, we can read off a tiling of an $m \times n$ grid by inspecting the colours assigned to the elements of the model. On the other hand, given any tiling for $\mathcal{T}$, we can construct a finite model of $\Phi_{\mathcal{T}}$ in the obvious way. We leave the detailed arguments to the reader.

The case of ω-words can be treated essentially in the same way. We just mark one element in a model, corresponding to the upper-right corner of the grid, with a special unary symbol, and relativize all our formulas to positions smaller than this element (marked with another fresh unary symbol). In effect, it is irrelevant what happens in the infinite fragment of a model starting in this marked element.

What is probably worth commenting is that in our undecidability proof we use the equivalence relation $\sim$ in a very limited way, actually not benefiting from its transitivity or symmetry. In fact, the transitivity of $\sim$ does not help, being rather an obstacle in our construction. ◀

5.2 Other variants

Both $\mathrm{FO}^2[<]$ and $\mathrm{FO}^2[+1]$ remain decidable when, besides $<$ or $+1$, the signature may contain other binary symbols, whose interpretation is not restricted ([15], [3]). We can easily see that this is not the case for F_1.

▶ **Theorem 14.** *The satisfiability problem for* $\mathrm{F}_1[+1]$ *and* $\mathrm{F}_1[<]$ *is undecidable when an additional uninterpreted binary relation is available.*

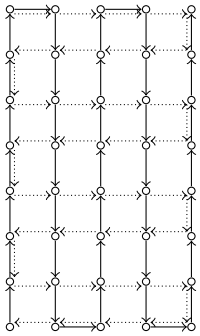


Figure 4 The grid-like structure used to show undecidability of $\mathrm{F}_1[+1_a, +1_b]$.

Actually, the proof is trivial and can be obtained even without using the linear order (just a simple grid axiomatization using a single binary predicate and some unary coordinate predicates). See [8] for a proof using two binary symbols.

There is a huge list of other, more specialized variations, which was studied in the context of FO^2, and which thus may also be considered here. One of the options is to have two linear orders rather than just one. The second linear order may be interpreted, e.g., as a comparison relation on data values. $\mathrm{FO}^2[+1_a, +1_b]$, the two-variable fragment accessing the linear orders through their successor relations only, is decidable in NExpTime [3]. Showing that a corresponding variant of F_1 is undecidable is again easy. We can define a grid-like structure using the first linear order to form a snake-like path as in the proof of Thm. 12 and the second to form another snake-like path, starting in the upper-left corner, ending in the lower-left corner and going horizontally through our grid, with steps down only on the borders. See Fig. 4. The required structure can be defined with help of some additional unary predicates. Since the details of the construction do not differ significantly from the details of the proof of Thm. 12 we omit them here.

▶ **Theorem 15.** *The satisfiability problem for* $\mathrm{F}_1[+1_a, +1_b]$ *is undecidable.*

6 Conclusion

This article is a starting point for investigations of the one-dimensional fragment of first-order logic over restricted classes of structures. We proved that $\mathrm{F}_1[<, +1]$ is expressively equivalent over words and ω-words to $\mathrm{FO}^2[<, +1]$, and that it also retains the complexity of the satisfiability problem of the latter. We argued on the other hand that some natural extensions of $\mathrm{F}_1[<, +1]$, whose two-variable counterparts remain decidable, become undecidable. One of such extensions are data words. Regarding the case of words we leave some open questions, the most important of which is if F_1 is more succinct than FO^2. Our working hypothesis is that it is true. A positive answer would be a good motivation for $\mathrm{F}_1[<, +1]$, showing that it allows to describe some properties of words not only in a more elegant, but simply also in a shorter way.

Our plan is also to check if the techniques developed in this paper can be used to show that the satisfiability problem of F_1 over trees retains the ExpSpace-complexity of FO^2 [1]. We suspect that it is true, even though this time the expressive power of both formalisms seems to differ: e.g., over unordered words, having access only to the descendant predicate, one can say in F_1 that there are at least three nodes satisfying a unary predicate P, such that none of them is a descendant of other, which is inexpressible in FO^2.

As mentioned in Section 5.2 there are a couple of specialized variations whose satisfiability may be considered. We suggest two of them here:

1. $F_1[<_a, <_b]$ (two linear orders accessed by *less then* predicates); in the case of FO^2 this variant is decidable at least over finite models [16].
2. $F_1[<, \sim]$ (data words, without access to the successor relation); this variant is NExpTime-complete for FO^2 [2]; moreover, in the case of FO^2 it remains decidable even if additional uninterpreted binary symbols are allowed [14]; of course, due to Thm. 14 we surely will not be able to lift the latter result to F_1.

References

1 Saguy Benaim, Michael Benedikt, Witold Charatonik, Emanuel Kieroński, Rastislav Lenhardt, Filip Mazowiecki, and James Worrell. Complexity of two-variable logic on finite trees. In *ICALP (2)*, pages 74–88, 2013. doi:10.1007/978-3-642-39212-2_10.

2 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27, 2011.

3 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and trees. In *LICS*, pages 73–82, 2013.

4 Witold Charatonik and Piotr Witkowski. Two-variable logic with counting and a linear order. In *Computer Science Logic*, volume 41 of *LIPIcs*, pages 631–647, 2015.

5 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory*. Perspectives in Mathematical Logic. Springer, 1995.

6 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.

7 Erich Grädel, Phokion Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

8 Lauri Hella and Antti Kuusisto. One-dimensional fragment of first-order logic. In *Advances in Modal Logic 10*, pages 274–293, 2014.

9 Emanuel Kieronski and Antti Kuusisto. Complexity and expressivity of uniform one-dimensional fragment with equality. In *Mathematical Foundations of Computer Science, Part I*, pages 365–376, 2014.

10 Emanuel Kieronski and Antti Kuusisto. Uniform one-dimensional fragments with one equivalence relation. In *Computer Science Logic*, volume 41 of *LIPIcs*, pages 597–615, 2015.

11 Emanuel Kieroński and Martin Otto. Small substructures and decidability issues for first-order logic with two variables. *Journal of Symbolic Logic*, 77:729–765, 2012.

12 Andreas Krebs, Kamal Lodaya, Paritosh Pandya, and Howard Straubing. Two-variable logic with a between predicate. In *LICS*, 2016.

13 Antti Kuusisto. On the uniform one-dimensional fragment. In *Proceedings of International Workshop on Description Logic*, 2016.

14 Angelo Montanari, Marco Pazzaglia, and Pietro Sala. Metric propositional neighborhood logic with an equivalence relation. In *TIME*, pages 49–58, 2014.

15 Martin Otto. Two-variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66:685–702, 2001.

16 Thomas Schwentick and Thomas Zeume. Two-variable logic with two order relations. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:15)2012.

17 Dana Scott. A decision method for validity of sentences in two variables. *Journal Symbolic Logic*, 27:477, 1962.

18 Larry J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, MIT, Cambridge, Massasuchets, USA, 1974.

19 Wieslaw Szwast and Lidia Tendera. FO^2 with one transitive relation is decidable. In *STACS*, pages 317–328, 2013.

Quine's Fluted Fragment is Non-Elementary*

Ian Pratt-Hartmann[1], Wiesław Szwast[2], and Lidia Tendera[3]

1 **School of Computer Science, University of Manchester, UK**
ipratt@cs.man.ac.uk
2 **Institute of Mathematics and Informatics, Opole University, Poland**
szwast@math.uni.opole.pl
2 **Institute of Mathematics and Informatics, Opole University, Poland**
tendera@math.uni.opole.pl

Abstract

We study the fluted fragment, a decidable fragment of first-order logic with an unbounded number of variables, originally identified by W.V. Quine. We show that the satisfiability problem for this fragment has non-elementary complexity, thus refuting an earlier published claim by W.C. Purdy that it is in NExpTime. More precisely, we consider, for all m greater than 1, the intersection of the fluted fragment and the m-variable fragment of first-order logic. We show that this sub-fragment forces $(m/2)$-tuply exponentially large models, and that its satisfiability problem is $(m/2)$-NExpTime-hard. We round off by using a corrected version of Purdy's construction to show that the m-variable fluted fragment has the m-tuply exponential model property, and that its satisfiability problem is in m-NExpTime.

1998 ACM Subject Classification F.4.1 Mathematical logic

Keywords and phrases Quine, fluted fragment, Purdy, non-elementary, satisfiability, decidability

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.39

1 Introduction

The fluted fragment, here denoted $\mathcal{FL}$, is a fragment of first-order logic in which, roughly speaking, the order of quantification of variables coincides with the order in which those variables appear as arguments of predicates. Fluted formulas arise naturally as first-order translations of quantified English sentences in which no quantifier-rescoping occurs, thus:

$$\begin{array}{l}\text{No student admires every professor}\\ \forall x_1(\text{student}(x_1) \to \neg\forall x_2(\text{prof}(x_2) \to \text{admires}(x_1, x_2)))\end{array} \tag{1}$$

$$\begin{array}{l}\text{No lecturer introduces any professor to every student}\\ \forall x_1(\text{lecturer}(x_1) \to \neg\exists x_2(\text{prof}(x_2)\wedge\ \forall x_3(\text{student}(x_3) \to \text{intro}(x_1, x_2, x_3)))).\end{array} \tag{2}$$

Furthermore, as was observed in [4], various standard translations of multi-modal logic into first-order logic are also easily seen to yield only fluted formulas. The origins of the fluted fragment can be traced to a paper given by W.V. Quine to the 1968 *International Congress of Philosophy* [11], in which the author defined what he called the *homogeneous m-adic formulas*. In these formulas, all predicates have the same arity m, and all atomic formulas have the same argument sequence $x_1, \ldots, x_m$. Boolean operators and quantifiers may be freely applied, except that the order of quantification must follow the order of arguments: a quantifier

* This work is supported by the Polish National Science Centre grant DEC-2013/09/B/ST6/01535 to W.S. and L.T., and by the EPSRC grant EP/K017438/1 to I.P-H.

© Ian Pratt-Hartmann, Wiesław Szwast, and Lidia Tendera;
licensed under Creative Commons License CC-BY

binding an occurrence of x_i may only be applied to a subformula in which all occurrences of $x_{i+1}, \ldots, x_m$ are already bound. Quine explained how Herbrand's decision procedure for monadic first-order logic easily extends to cover all homogeneous m-adic formulas. The term *fluted logic* first appears (to the present authors' knowledge) in [13], where the restriction that all predicates have the same arity is abandoned, a relaxation which, according to Quine, does not affect the proof of decidability of satisfiabilty. It seems that the allusion is architectural rather than musical: we are invited to think of arguments of predicates as being 'lined up' in columns. Quine's motivation for defining the fluted fragment was to locate the boundary of decidability in the context of his reconstruction of first-order logic in terms of *predicate-functors*, which Quine himself described as a 'modification of Bernays' modification of Tarski's cylindrical algebra' [12, p. 299]. Specifically, the fluted fragment can be identified by dropping from full predicate functor logic those functors associated with the permutation and identification of variables, while retaining those concerned with cylindrification and Boolean combination.

Notwithstanding its predicate-functorial lineage, the fluted fragment has, as we shall see, a completely natural characterization within the standard régime of bound variable quantification, and thus constitutes an interesting fragment of first-order logic in its own right. In fact, $\mathcal{FL}$ overlaps in expressive power with various other such fragments. For example, *Boolean modal logic* [5] maps, under the standard first-order translation, to $\mathcal{FL}$ – in fact, to $\mathcal{FL}^2$, the fluted fragment restricted to just two variables. (Thus, $\mathcal{FL}^2$ in effect subsumes the description logic $\mathcal{ALC}$.) On the other hand, even $\mathcal{FL}^2$ is not contained within the so-called *guarded fragment* of first-order logic [1]: the formula (1), for example, is not equivalent to any guarded formula. A more detailed comparison of the fluted fragment to other familiar decidable fragments can be found in [4].

Noah [6] pointed out, however, that – contrary to Quine's assertion – Herbrand's technique does not obviously extend from homogeneous m-adic logic to the fluted fragment, and consequently, the decidability of the satisfiability problem for the latter should be regarded as open. This problem – together with the corresponding problems for various extensions of the fluted fragment – was considered in a series of papers in the 1990s by W.C. Purdy [7, 8, 9, 10]. The decidability of $\mathcal{FL}$ is proved in [8], while in [10] it is claimed (Corollary 10) that this fragment has the exponential-sized model property: if a fluted formula φ is satisfiable, then it is satisfiable over a domain of size bounded by an exponential function of the number of symbols in φ. Purdy concluded (Theorem 13) that the satisfiability problem for $\mathcal{FL}$ is NExpTime-complete.

These latter claims are false. In the sequel, we show that, for $m \geq 2$, the fluted fragment restricted to just m variables, denoted $\mathcal{FL}^m$, can force models of $(\lfloor m/2 \rfloor)$-tuply exponential size, and that its satisfiability problem is $(\lfloor m/2 \rfloor)$-NExpTime-hard. It follows that there is no elementary bound on the size of models of satisfiable fluted formulas, and that the satisfiability problem for $\mathcal{FL}$ is non-elementary. On the other hand, we also show that any satisfiable formula of the m-variable fluted fragment has a model of m-tuply exponential size, so that the satisfiability problem for this sub-fragment is contained in m-NExpTime. Thus, $\mathcal{FL}$ has the finite model property, and its satisfiability (= finite satisfiability) problem is decidable, but not elementary. Note that the above complexity bounds for $\mathcal{FL}^m$ leave a gap of a factor of 2.

We mention at this point another incorrect claim by Purdy concerning an extension of the fluted fragment. In [9], the author considers what he calls *extended fluted logic*, in which, in addition to the usual predicate functors of fluted logic, we have an *identity functor* (essentially: the equality predicate), *binary conversion* (the ability to exchange arguments

in binary atomic formulas) and *functions* (the requirement that certain specified binary predicates be interpreted as the graph of a function.) Purdy claims (Corollary 19, p. 1460) that EFL has the *finite model property*: if a formula of this fragment is satisfiable, then it is satisfiable over a finite domain. But EFL evidently contains the formula

$$\forall x_1 \forall x_2 (r(x_1, x_2) \to f(x_1, x_2)) \wedge \exists x_1 \forall x_2 \neg r(x_1, x_2) \wedge \forall x_1 \exists x_2 r(x_2, x_1),$$

where f is required to be interpreted as the graph of a binary function; and this is an axiom of infinity. In view of these observations, it seems only prudent to treat Purdy's series of articles with caution. We also mention that an independent decision procedure for the fluted fragment – based on resolution theorem-proving – was presented in [15]. No complexity bounds are given there. Moreover, that paper omits detailed proofs, and these have, to the authors' knowledge, never been published.

The structure of this paper is as follows. Section 2 gives some basic definitions. In Section 3, we show that formulas of $\mathcal{FL}^{2m}$ can force models of m-tuply exponential size, and indeed that the satisfiability problem for $\mathcal{FL}^{2m}$ is m-NExpTime-hard, thus disproving the results claimed in [10]. In Section 4, we show how some of the constructions appearing in Purdy's paper can nevertheless be recycled to give a proof that the fluted fragment with m variables does indeed have the finite model property, and that its satisfiability problem is in m-NExpTime. This proof is shorter and more perspicuous than the original argument for the decidability of $\mathcal{FL}$ given in [8], and, moreover, yields detailed complexity information.

2 Preliminaries

Let $\bar{x}_\omega = x_1, x_2, \ldots$ be a fixed sequence of variables. We define the sets of formulas $\mathcal{FL}^{[k]}$ (for $k \geq 0$) by structural induction as follows: (i) any atom $p(x_\ell, \ldots, x_k)$, where $x_\ell, \ldots, x_k$ is a contiguous subsequence of $\bar{x}_\omega$, is in $\mathcal{FL}^{[k]}$; (ii) $\mathcal{FL}^{[k]}$ is closed under boolean combinations; (iii) if φ is in $\mathcal{FL}^{[k+1]}$, then $\exists x_{k+1}\varphi$ and $\forall x_{k+1}\varphi$ are in $\mathcal{FL}^{[k]}$. The set of *fluted formulas* is defined as $\mathcal{FL} = \bigcup_{k\geq 0} \mathcal{FL}^{[k]}$. A *fluted sentence* is a fluted formula over an empty set of variables, i.e. an element of $\mathcal{FL}^{[0]}$. Thus, when forming Boolean combinations in the fluted fragment, all the combined formulas must have as their free variables some suffix of some prefix $x_1, \ldots, x_k$ of $\bar{x}_\omega$; and when quantifying, only the last variable in this sequence may be bound, as illustrated by the fluted sentences in (1) and (2). Note that, in this paper, we consider only purely relational signatures.

Denote by $\mathcal{FL}^m$ the sub-fragment of $\mathcal{FL}$ consisting of those formulas featuring at most m variables, free or bound. Do not confuse $\mathcal{FL}^m$ (the set of fluted formulas with m variables, free or bound) with $\mathcal{FL}^{[m]}$ (the set of fluted formulas with m *free* variables). Thus, the formulas in (1) and (2) are in $\mathcal{FL}^{[0]}$; however (1) is in $\mathcal{FL}^m$ only for $m \geq 2$, and (2) is in $\mathcal{FL}^m$ only for $m \geq 3$. All formulas occurring in the remainder of the paper will be fluted.

In the sequel, we employ standard concepts and notation from first-order logic. Structures are denoted by Gothic capital letters and their domains by the corresponding Roman capitals. If $\mathfrak{A}$ is a structure, $\varphi(x_1, \ldots, x_k)$ a formula of $\mathcal{FL}^{[k]}$, and $\bar{a}$ a k-tuple of elements of A, then we write $\mathfrak{A} \models \varphi[\bar{a}]$ to indicate that $\bar{a}$ satisfies $\varphi(x_1, \ldots, x_k)$ in $\mathfrak{A}$; in the case where φ is a fluted sentence and $\mathfrak{A}$ is a model of φ, we write simply $\mathfrak{A} \models \varphi$. If φ is a formula, we write $\|\varphi\|$ to denote the number of symbols in φ. We use $\pm\varphi$ to stand either for φ or $\neg\varphi$, with multiple occurrences of $\pm$ in displayed material resolved *uniformly*: thus, for example $\pm p_i^1(x_1, x_2) \to \pm p_i^1(x_2)$ stands for a *pair* (not a quartet) of formulas, namely, $p_i^1(x_1, x_2) \to p_i^1(x_2)$ and $\neg p_i^1(x_1, x_2) \to \neg p_i^1(x_2)$.

3 Lower bound

In this section, we establish lower complexity bounds for the fluted fragment, which we express using the *tetration function* $\mathfrak{t}(k,n)$, defined, for $n,k \geq 0$, by induction as follows:

$$\begin{aligned}\mathfrak{t}(0,n) &= n\\ \mathfrak{t}(k+1,n) &= 2^{\mathfrak{t}(k,n)}.\end{aligned}$$

Thus, $\mathfrak{t}(1,n) = 2^n$, $\mathfrak{t}(2,n) = 2^{2^n}$, and so on. Theorem 1 shows that an $\mathcal{FL}^{2m}$-formula of size $O(n^2)$ can force models of size at least $\mathfrak{t}(m,n)$, thus contradicting Corollary 10 of [10]. Theorem 2 shows that the satisfiability problem for $\mathcal{FL}^{2m}$ is m-NExpTime-hard, thus contradicting Theorem 11 of [10].

As a preliminary, for any $z \geq 0$, we take the (*canonical*) *representation* of any integer n in the range $(0 \leq j < 2^z)$ to be the bit-string $\bar{s} = s_{z-1}, \ldots, s_0$ of length z, where $n = \sum_{i=0}^{z-1} s_i \cdot 2^i$. (Thus, s_0 is the least significant bit.) Where z is clear from context, this representation is unique. Observe that, if, in addition, an integer n' in the same range is represented by $s'_{z-1}, \ldots, s'_0$, then $n' = n - 1 \mod 2^z$ if and only if, for all i $(0 \leq i < z)$:

$$s'_i = \begin{cases} 1 - s_i & \text{if, for all } j \ (0 \leq j < i),\ s_j = 0;\\ s_i & \text{otherwise.}\end{cases}$$

This simple observation – effectively, the algorithm for decrementing an integer represented in binary – will feature at various points in the proof of the following theorem.

▶ **Theorem 1.** *For all $m \geq 1$, there exists a sequence of satisfiable sentences $\{\varphi_n\}_{n\in\mathbb{N}} \in \mathcal{FL}^{2m}$ such that $\|\varphi_n\|$ grows polynomially with m and n (and indeed quadratically in n for fixed m), but the smallest satisfying model of φ_n has at least $\mathfrak{t}(m,n)$ elements. Hence, there is no elementary bound on the size of models of satisfiable sentences in $\mathcal{FL}$.*

Proof. Fix positive integers m and n. Consider a signature $\Sigma_{m,n}$ featuring:

- unary predicates $p_0, \ldots, p_{n-1}$;
- for all k in the range $1 \leq k \leq m$, a unary predicate int_k;
- for all k in the range $1 \leq k < m$, binary predicates in_k, out_k.

(We shall add further predicates to $\Sigma_{m,n}$ in the course of the proof.) When working within a particular structure, we call any element satisfying the unary predicate int_k in that structure a *k-integer*. Each k-integer, b, will be associated with an integer value, $\mathrm{val}_k(b)$, between 0 and $\mathfrak{t}(k,n) - 1$. For $k = 1$, this value will be encoded by b's satisfaction of the unary predicates $p_0, \ldots, p_{n-1}$. Specifically, for any 1-integer b, define $\mathrm{val}_1(b)$ to be the integer canonically represented by the n-element bit-string $s_{n-1}, \ldots, s_0$, where, for all i $(0 \leq i < n)$,

$$s_i = \begin{cases} 1 & \text{if } \mathfrak{A} \models p_i[b];\\ 0 & \text{otherwise.}\end{cases}$$

On the other hand, if b is a $(k+1)$-integer $(k \geq 1)$, then $\mathrm{val}_{k+1}(b)$ will be encoded by how b is related to the various k-integers via the predicate in_k. Specifically, for any k $(1 \leq k < m)$ and any $(k+1)$-integer b, define $\mathrm{val}_{k+1}(b)$ to be the integer canonically represented by the bit-string $s_{N-1}, \ldots, s_0$ of length $N = \mathfrak{t}(k,n)$ where, for all i $(0 \leq i < N)$,

$$s_i = \begin{cases} 1 & \text{if } \mathfrak{A} \models \mathrm{in}_k[a,b] \text{ for some } (k)\text{-integer } a \text{ s.t. } \mathrm{val}_k(a) = i;\\ 0 & \text{otherwise.}\end{cases}$$

We shall be interested in the case where $\mathfrak{A}$ satisfies the following two properties, for all k $(1 \leq k \leq m)$.

***k*-covering:** The function $\mathrm{val}_k : \mathrm{int}_k^{\mathfrak{A}} \to [0, \mathfrak{t}(k,n) - 1]$ is surjective.
***k*-harmony:** If $k > 1$, then, for all k-integers b and all $(k-1)$-integers a, a' in $\mathfrak{A}$ such that $\mathrm{val}_{k-1}(a) = \mathrm{val}_{k-1}(a')$, we have $\mathfrak{A} \models \mathrm{in}_{k-1}[a,b] \Leftrightarrow \mathfrak{A} \models \mathrm{out}_{k-1}[b,a']$.

If $k < m$, k-covering ensures that, when we want to know what the ith bit in the canonical binary representation of a $(k+1)$-integer b is (where $0 \leq i < \mathfrak{t}(k,n)$), then *there exists* a k-integer a such that $\mathrm{val}_k(a) = i$, and for which we can ask whether $\mathfrak{A} \models \mathrm{in}_k[a,b]$. Conversely, $(k+1)$-harmony ensures that, if there are *many* such a, then it does not matter which one we consult. For if $\mathrm{val}_k(a) = \mathrm{val}_k(a')$, then by two applications of $(k+1)$-harmony, $\mathfrak{A} \models \mathrm{in}_k[a,b] \Leftrightarrow \mathfrak{A} \models \mathrm{out}_k[b,a] \Leftrightarrow \mathfrak{A} \models \mathrm{in}_k[a',b]$.

The proof proceeds by writing a satisfiable $\mathcal{FL}^{2m}$-formula $\Phi_{m,n}$ in the signature $\Sigma_{m,n}$ such that any model $\mathfrak{A} \models \Phi_{m,n}$ satisfies k-covering and k-harmony for all k $(1 \leq k \leq m)$. It follows from m-covering that $|A| \geq \mathfrak{t}(m,n)$, proving the theorem. The signature $\Sigma_{m,n}$ will feature several auxiliary predicates. In particular, we take $\Sigma_{m,n}$ to contain: (i) the unary predicates $\mathrm{zero}_1, \ldots, \mathrm{zero}_m$; (ii) the binary predicates $\mathrm{pred}_{1,0}, \ldots, \mathrm{pred}_{m,0}$; and (iii) the ternary predicates $\mathrm{pred}_{1,1}, \ldots, \mathrm{pred}_{m-1,1}$. Furthermore, we take $\Sigma_{m,n}$ to contain, for all k $(1 \leq k \leq m)$ and all ℓ $(0 \leq \ell \leq 2(m-k))$, the $(\ell+2)$-ary predicate $\mathrm{eq}_{k,\ell}$. Observe that $\Sigma_{m,n}$ does not contain the predicate $\mathrm{pred}_{m,1}$. Observe also that, as k increases from 1 to m, the maximal value of the index ℓ in the predicates $\mathrm{eq}_{k,\ell}$ decreases, in steps of 2, from $2m-2$ down to 0; hence the maximal arity of these predicates decreases from $2m$ to 2.

Any model $\mathfrak{A} \models \Phi_{m,n}$ will be guaranteed to satisfy the following properties for all k $(1 \leq k \leq m)$ concerning the interpretation of these predicates.
***k*-zero:** For all k-integers b, $\mathfrak{A} \models \mathrm{zero}_k[b] \Leftrightarrow \mathrm{val}_k(b) = 0$.
***k*-equality:** For all ℓ $(0 \leq \ell \leq 2(m-k))$, all k-integers b, b' and all ℓ-tuples of elements $\bar{c}$, $\mathfrak{A} \models \mathrm{eq}_{k,\ell}[b, \bar{c}, b'] \Leftrightarrow \mathrm{val}_k(b) = \mathrm{val}_k(b')$.
***k*-predecessor:** For all ℓ $(0 \leq \ell \leq \min(m-k, 1))$, all k-integers b, b' and all ℓ-tuples of elements $\bar{c}$, $\mathfrak{A} \models \mathrm{pred}_{k,\ell}[b, \bar{c}, b'] \Leftrightarrow \mathrm{val}_k(b') = \mathrm{val}_k(b) - 1$, modulo $\mathfrak{t}(k,n)$.

The bounds on ℓ in the property k-predecessor amount to saying that ℓ takes values 0 or 1, except when $k = m$, in which case it takes only the value 0. (Recall that $\Sigma_{m,n}$ does not feature the predicate $\mathrm{pred}_{m,1}$.)

Thus, $\mathrm{zero}_k(x_1)$ can be read as "x_1 is zero", $\mathrm{pred}_{k,0}(x_1, x_2)$, as "$x_2$ is the predecessor of x_1", and $\mathrm{pred}_{k,1}(x_1, x_2, x_3)$ as "x_3 is the predecessor of x_1". Notice that, in the latter case, the argument x_2 is semantically inert. Similarly, $\mathrm{eq}_{k,\ell}(x_1, \ldots, x_{\ell+2})$ can be read as "x_1 is equal to $x_{\ell+2}$", with the ℓ arguments $x_2, \ldots, x_{\ell+1}$ again semantically inert. When naming predicates, we employ the convention that the first subscript, k, serves as a reminder that its primary arguments are typically assumed to be k-integers; the second subscript, ℓ, indicates that ℓ (possibly 0) semantically inert arguments have been inserted between the primary arguments.

We suppose that $\mathfrak{A} \models \Phi_{m,n}$, and establish the properties k-covering, k-harmony, k-zero, k-equality and k-predecessor for all k $(1 \leq k \leq m)$ by induction on k. For ease of reading, we introduce the various conjuncts of $\Phi_{m,n}$ as they are required. Appeals to the inductive hypothesis are indicated by the initials IH.

Base case ($k = 1$): Let b be a 1-integer, and recall that $\mathrm{val}_1(b)$ is defined by b's satisfaction of the predicates $p_0, \ldots, p_{n-1}$. We proceed to secure the properties required for the base case of the induction. The property 1-harmony is trivially satisfied. We secure 1-zero by adding to $\Phi_{m,n}$ the conjunct

$$\forall x_1 (\mathrm{int}_1(x_1) \to (\mathrm{zero}_1(x_1) \leftrightarrow \bigwedge_{i=0}^{n-1} \neg p_i(x_1))). \tag{Φ_1}$$

Thus, if a is a 1-integer, $\mathfrak{A} \models \mathrm{zero}_1[b] \Leftrightarrow \mathrm{val}_1(b) = 0$. To do the same for 1-predecessor and 1-equality, we proceed as follows. Letting $L = 2m-1$, we add to $\Sigma_{m,n}$ an $(\ell+1)$-ary predicate, p_i^ℓ, for all i $(0 \leq i < n)$ and all ℓ $(0 \leq \ell \leq L)$, and we add to $\Phi_{m,n}$ the corresponding pair of conjuncts

$$\bigwedge_{i=0}^{n-1} \bigwedge_{\ell=0}^{L} \forall x_1(\mathrm{int}_1(x_1) \wedge \pm p_i(x_1) \rightarrow \forall x_2 \cdots \forall x_{\ell+1} \pm p_i^\ell(x_1, \ldots, x_{\ell+1})). \tag{Φ_2}$$

Note that this really is a *pair* of formulas: the two occurrences of the $\pm$ sign must be resolved in the same way.

Then, for any 1-integer b and any ℓ-tuple $\bar{c}$ from A,

$$\mathfrak{A} \models p_i^\ell[b, \bar{c}] \Leftrightarrow \mathfrak{A} \models p_i[b]. \tag{3}$$

In effect, the conjuncts (Φ_2) append semantically inert arguments to each of the predicates p_i. This technique will be helpful at several points in the sequel, and we employ the convention that a superscript ℓ on a predicate letter indicates that the corresponding undecorated predicate has had ℓ semantically inert arguments *appended* to its primary arguments. Note that p_i^0 is simply equivalent to p_i.

Now we can secure the property 1-equality. For all ℓ $(0 \leq \ell < 2m-2)$, let $\varepsilon_{1,\ell}(x_1, \ldots, x_{\ell+2})$ abbreviate the formula: $\bigwedge_{i=0}^{n-1}(p_i^{\ell+1}(x_1, \ldots, x_{\ell+2}) \leftrightarrow p_i(x_{\ell+2}))$. We see from (3) that $\varepsilon_{1,\ell}(x_1, \ldots, x_{\ell+2})$ in effect states that (for x_1 and $x_{\ell+2}$ 1-integers) the values of x_1 and $x_{\ell+2}$ are identical. We therefore add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{2m-2} \forall x_1(\mathrm{int}_1(x_1) \rightarrow \forall x_2 \cdots \forall x_{\ell+2}(\mathrm{int}_1(x_{\ell+2}) \rightarrow \mathrm{eq}_{1,\ell}(x_1, \ldots, x_{\ell+2}) \leftrightarrow \varepsilon_{1,\ell}(x_1, \ldots, x_{\ell+2}))). \tag{Φ_3}$$

Thus, for any 1-integers b, b' in $\mathfrak{A}$ and any ℓ-tuple $\bar{c}$ from A $(0 \leq \ell < 2m-2)$, $\mathfrak{A} \models \mathrm{eq}_{1,\ell}[b, \bar{c}, b'] \Leftrightarrow \mathrm{val}_1(b) = \mathrm{val}_1(b')$.

Turning to the property 1-predecessor, assume for the moment that $m > 1$, so that the predicates $\mathrm{pred}_{1,0}$ and $\mathrm{pred}_{1,1}$ are both in $\Sigma_{m,n}$. For $0 \leq \ell \leq 1$, let $\pi_{1,\ell}(x_1, \ldots, x_{\ell+2})$ abbreviate the formula

$$\bigwedge_{i=0}^{n-1} ([\bigwedge_{j=0}^{i-1} \neg p_j^{\ell+1}(x_1, \ldots, x_{\ell+2}) \rightarrow (p_i^{\ell+1}(x_1, \ldots, x_{\ell+2}) \leftrightarrow \neg p_i(x_{\ell+2}))] \wedge [\bigvee_{j=0}^{i-1} p_j^{\ell+1}(x_1, \ldots, x_{\ell+2}) \rightarrow (p_i^{\ell+1}(x_1, \ldots, x_{\ell+2}) \leftrightarrow p_i(x_{\ell+2}))]).$$

From our preliminary remarks on the canonical representations of numbers by bit-strings, we see that $\pi_{1,\ell}(x_1, \ldots, x_{\ell+2})$ codes the statement that (for x_1 and $x_{\ell+2}$ 1-integers) the value of $x_{\ell+2}$ is one less than that of $x_1 \mod 2^n$. We then add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{1} \forall x_1(\mathrm{int}_1(x_1) \rightarrow \forall x_2 \cdots \forall x_{\ell+2}(\mathrm{int}_1(x_{\ell+2}) \rightarrow (\mathrm{pred}_{1,\ell}(x_1, \ldots, x_{\ell+2}) \leftrightarrow \pi_{1,\ell}(x_1, \ldots, x_{\ell+2})))), \tag{Φ_4}$$

securing the property 1-predecessor, as required.

If, on the other hand, $m = 1$, we proceed in the same way, except that we add only the conjunct of (Φ_4) with index $\ell = 0$; this suffices to satisfy the property 1-predecessor. Observe that care is required in this case, because $\Sigma_{1,n}$ does not feature the ternary predicate $\mathrm{pred}_{1,1}$ – indeed, it features no ternary predicates at all.

Finally, to secure 1-covering, we add to $\Phi_{m,n}$ the conjuncts

$$\exists x_1(\mathrm{int}_1(x_1) \wedge \mathrm{zero}_1(x_1)) \tag{Φ_5}$$

$$\forall x_1(\mathrm{int}_1(x_1) \to \exists x_2(\mathrm{int}_1(x_2) \wedge \mathrm{pred}_{1,0}(x_1, x_2))). \tag{Φ_6}$$

Observe that (Φ_6) features only $\mathrm{pred}_{1,0}$, and not $\mathrm{pred}_{1,1}$, and so does not stray outside $\Sigma_{m,n}$, even when $m = 1$.

Inductive case: This case arises only if $m \geq 2$. Assume that $\mathrm{val}_k : \mathrm{int}_k^{\mathfrak{A}} \to [0, \mathfrak{t}(k, n) - 1]$ satisfies the properties of k-harmony, k-zero, k-predecessor, k-covering and k-equality. We show, by adding appropriate conjuncts to $\Phi_{m,n}$, that these properties hold with k replaced by $k + 1$.

For $(k + 1)$-harmony, we add to $\Phi_{m,n}$ the following pair of conjuncts:

$$\begin{aligned}\forall x_1(\mathrm{int}_k(x_1) \to \forall x_2(\mathrm{int}_{k+1}(x_2) \wedge \pm\mathrm{in}_k(x_1, x_2) \to \\ \forall x_3(\mathrm{int}_k(x_3) \wedge \mathrm{eq}_{k,1}(x_1, x_2, x_3) \to \pm\mathrm{out}_k(x_2, x_3))))).\end{aligned} \tag{Φ_7}$$

If a, a' are k-integers such that $\mathrm{val}_k(a) = \mathrm{val}_k(a')$, and b is any $(k + 1)$-integer, then, by k-equality (IH), $\mathfrak{A} \models \mathrm{eq}_{k,1}[a, b, a']$, whence (Φ_7) evidently secures $(k + 1)$-harmony.

We remind ourselves at this point of the role of $(k + 1)$-harmony in the subsequent argument, and, in particular, on its relationship to k-covering. Let b be a $(k + 1)$-integer, and recall that $\mathrm{val}_{k+1}(b)$ is defined by b's satisfaction of the predicates in_k in relation to the various k-integers in $\mathfrak{A}$. By k-covering (IH), for all i $(0 \leq i < \mathfrak{t}(k, n))$, there *is* a k-integer a with $\mathrm{val}_k(a) = i$; and by $(k + 1)$-harmony (just established), all such k-integers a agree on what the ith bit in $\mathrm{val}_{k+1}(b)$ should be.

To secure $(k + 1)$-zero, we add to $\Phi_{m,n}$ the conjunct

$$\forall x_1(\mathrm{int}_{k+1}(x_1) \to (\mathrm{zero}_{k+1}(x_1) \leftrightarrow \forall x_2(\mathrm{int}_k(x_2) \to \neg\mathrm{out}_k(x_1, x_2))))). \tag{Φ_8}$$

From $(k + 1)$-harmony and (Φ_8) we see that, for all $(k + 1)$-integers b, $\mathfrak{A} \models \mathrm{zero}_{k+1}[b] \Leftrightarrow (\mathrm{val}_{k+1}(b) = 0)$. For if there were any k-integer a, such that $\mathfrak{A} \models \mathrm{in}_k[a, b]$, then we would have $\mathfrak{A} \models \mathrm{out}_k[b, a]$.

Establishing the property $(k + 1)$-predecessor is more involved. We add to $\Sigma_{m,n}$ binary predicates $\mathrm{in}_k^{\triangleleft}$, $\mathrm{out}_k^{\triangleleft}$. The idea is that, for any k-integer a and any $(k + 1)$-integer b:

$$\mathfrak{A} \models \mathrm{in}_k^{\triangleleft}[a, b] \Leftrightarrow (\text{for any } k\text{-integer } a', \mathrm{val}_k(a') < \mathrm{val}_k(a) \Rightarrow \mathfrak{A} \not\models \mathrm{in}_k[a', b]); \tag{4}$$

$$\mathfrak{A} \models \mathrm{in}_k^{\triangleleft}[a, b] \Leftrightarrow \mathfrak{A} \models \mathrm{out}_k^{\triangleleft}[b, a]. \tag{5}$$

Condition (4) allows us to read $\mathrm{in}_k^{\triangleleft}(x_1, x_2)$ as "all the bits in the value of the $(k + 1)$-integer x_2 whose index is less than the value of the k-integer x_1 are zero." Condition (5) is somewhat analogous to $(k + 1)$-harmony.

Securing Condition (5) is easy. We add to $\Phi_{m,n}$ the pair of conjuncts

$$\begin{aligned}\forall x_1(\mathrm{int}_k(x_1) \to \forall x_2(\mathrm{int}_{k+1}(x_2) \wedge \pm\mathrm{in}_k^{\triangleleft}(x_1, x_2) \to \\ \forall x_3(\mathrm{int}_k(x_3) \wedge \mathrm{eq}_{k,1}(x_1, x_2, x_3) \to \pm\mathrm{out}_k^{\triangleleft}(x_2, x_3))))).\end{aligned} \tag{Φ_9}$$

For let a be a k-integer and b a $(k + 1)$-integer. By the property k-equality (IH), $\mathfrak{A} \models \mathrm{eq}_{k,1}[a, b, a]$, whence (5) follows.

Securing Condition (4) is harder. We first add to $\Sigma_{m,n}$ a binary predicate zero_k^1, which appends one semantically inert argument to the unary predicate zero_k. That is, we add to $\Phi_{m,n}$ the pair of conjuncts

$$\forall x_1(\mathrm{int}_k(x_1) \wedge \pm\mathrm{zero}_k(x_1) \to \forall x_2 \pm \mathrm{zero}_k^1(x_1, x_2)). \tag{Φ_{10}}$$

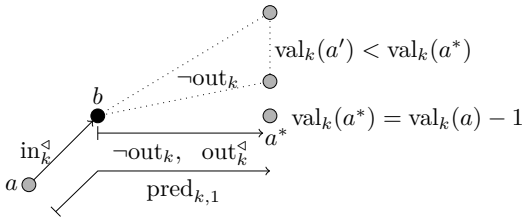


Figure 1 Fixing the interpretation of $\mathrm{in}_k^{\triangleleft}$.

We can then secure (4) by adding to $\Phi_{m,n}$ the conjunct

$$\forall x_1(\mathrm{int}_k(x_1) \rightarrow \forall x_2(\mathrm{int}_{k+1}(x_2) \rightarrow (\mathrm{in}_k^{\triangleleft}(x_1,x_2) \leftrightarrow (\mathrm{zero}_k^1(x_1,x_2) \vee \forall x_3(\mathrm{int}_k(x_3) \wedge \mathrm{pred}_{k,1}(x_1,x_2,x_3) \rightarrow (\mathrm{out}_k^{\triangleleft}(x_2,x_3) \wedge \neg\mathrm{out}_k(x_2,x_3)))))))). \quad (\Phi_{11})$$

To see this, we perform a subsidiary induction on the quantity $\mathrm{val}_k(a)$. Let a be any k-integer and b any $(k+1)$-integer. For the base case, suppose $\mathrm{val}_k(a) = 0$. Then $\mathfrak{A} \models \mathrm{zero}_k[a]$ by the property k-zero (IH), whence $\mathfrak{A} \models \mathrm{zero}_k^1[a,b]$ by (Φ_{10}), whence $\mathfrak{A} \models \mathrm{in}_k^{\triangleleft}[a,b]$ by (Φ_{11}). For the inductive step, suppose that $\mathrm{val}_k(a) > 0$; thus, by k-zero again, $\mathfrak{A} \not\models \mathrm{zero}_k[a]$. Assume first that $\mathfrak{A} \models \mathrm{in}_k^{\triangleleft}[a,b]$. By k-covering (IH), we may pick some k-integer a^* with $\mathrm{val}_k(a^*) = \mathrm{val}_k(a) - 1$. By k-predecessor (IH), setting $\ell = 1$, $\mathfrak{A} \models \mathrm{pred}_{k,1}[a,b,a^*]$, whence, taking x_1, x_2 and x_3 in (Φ_{11}) to be a, b and a^*, respectively, $\mathfrak{A} \models \mathrm{out}_k^{\triangleleft}[b,a^*]$ and $\mathfrak{A} \not\models \mathrm{out}_k[b,a^*]$. The situation is illustrated in Fig. 1. Applying the subsidiary inductive hypothesis, it follows from (4) and (5), with a replaced by a^*, that for any k-integer a' with $\mathrm{val}_k(a') < \mathrm{val}_k(a^*)$, $\mathfrak{A} \not\models \mathrm{in}_k[a',b]$. Moreover, by $(k+1)$-harmony (just established), $\mathfrak{A} \not\models \mathrm{out}_k[b,a^*]$ implies that, for any k-integer a' with $\mathrm{val}_k(a') = \mathrm{val}_k(a^*)$, $\mathfrak{A} \not\models \mathrm{in}_k[a',b]$. Thus, for any k-integer a', $\mathrm{val}_k(a') < \mathrm{val}_k(a) \Rightarrow \mathfrak{A} \not\models \mathrm{in}_k[a',b]$. Conversely, suppose that $\mathfrak{A} \not\models \mathrm{in}_k^{\triangleleft}[a,b]$. Then, from (Φ_{11}), there exists some k-integer a^* such that $\mathrm{pred}_{k,1}[a,b,a^*]$, but either $\mathfrak{A} \not\models \mathrm{out}_k^{\triangleleft}[b,a^*]$ or $\mathfrak{A} \models \mathrm{out}_k[b,a^*]$. By k-predecessor (IH), again setting $\ell = 1$, $\mathrm{val}_k(a^*) = \mathrm{val}_k(a) - 1$, and hence, applying the subsidiary inductive hypothesis, (4) and (5) ensure that, if $\mathfrak{A} \not\models \mathrm{out}_k^{\triangleleft}[b,a^*]$, then, for some k-integer a' with $\mathrm{val}_k(a') < \mathrm{val}_k(a^*) < \mathrm{val}_k(a)$, $\mathfrak{A} \models \mathrm{in}_k[a',b]$. On the other hand, by k-equality and $(k+1)$-harmony, $\mathfrak{A} \models \mathrm{out}_k[b,a^*]$ implies $\mathfrak{A} \models \mathrm{in}_k[a^*,b]$. Either way, there exists a k-integer a' such that $\mathrm{val}_k(a') < \mathrm{val}_k(a)$, but $\mathfrak{A} \models \mathrm{in}_k[a',b]$. This completes the (subsidiary) induction, and establishes (4).

Having fixed the interpretation of $\mathrm{in}_k^{\triangleleft}$, we proceed to secure the property $(k+1)$-predecessor. Assume first that $k+1 < m$. We add to $\Sigma_{m,n}$ the predicates in_k^2, in_k^3, $\mathrm{in}_k^{\triangleleft 2}$, $\mathrm{in}_k^{\triangleleft 3}$ and we add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{1} \forall x_1(\mathrm{int}_k(x_1) \rightarrow \forall x_2(\mathrm{int}_{k+1}(x_2) \wedge \pm\mathrm{in}_k(x_1,x_2) \rightarrow \forall x_3 \cdots \forall x_{\ell+4} \pm \mathrm{in}_k^{\ell+2}(x_1,\ldots,x_{\ell+4}))), \quad (\Phi_{12})$$

$$\bigwedge_{\ell=0}^{1} \forall x_1(\mathrm{int}_k(x_1) \rightarrow \forall x_2(\mathrm{int}_{k+1}(x_2) \wedge \pm\mathrm{in}_k^{\triangleleft}(x_1,x_2) \rightarrow \forall x_3 \cdots \forall x_{\ell+4} \pm \mathrm{in}_k^{\triangleleft \ell+2}(x_1,\ldots,x_{\ell+4})), \quad (\Phi_{13})$$

fixing these predicates to be the result of adding either 2 or 3 semantically inert arguments to in_k and $\mathrm{in}_k^{\triangleleft}$, as indicated by the superscripts.

In the sequel, we shall employ a formula denoted $\varrho_{k,0}(x_1,\ldots,x_4)$, which will be of interest where x_1 and x_4 are k-integers *with equal values*, while x_2 and x_3 are $(k+1)$-integers. Intuitively this formula says: "the $\mathrm{val}_k(x_1)$th – equivalently, $\mathrm{val}_k(x_4)$th – digits of x_2 and x_3 are as they should be if $\mathrm{val}_{k+1}(x_3) = \mathrm{val}_{k+1}(x_2) - 1 \mod \mathfrak{t}(k+1,n)$." Similarly, we shall employ the formula $\varrho_{k,1}(x_1,\ldots,x_5)$ which will be of interest where x_1 and x_5 are k-integers with equal values, while x_2 and x_4 are $(k+1)$-integers. Intuitively this formula says: "the $\mathrm{val}_k(x_1)$th – equivalently, $\mathrm{val}_k(x_5)$th – digits of x_2 and x_4 are as they should be if $\mathrm{val}_{k+1}(x_4) = \mathrm{val}_{k+1}(x_2) - 1 \mod \mathfrak{t}(k+1,n)$." Note that, in the latter case, x_3 is a dummy variable. Formally, for $0 \leq \ell \leq 1$, define $\varrho_{k,\ell}(x_1,\ldots,x_{\ell+4})$ to be the formula

$$[\mathrm{in}_k^{\lhd \ell+2}(x_1,\ldots,x_{\ell+4}) \to (\mathrm{in}_k^{\ell+2}(x_1,\ldots,x_{\ell+4}) \leftrightarrow \neg\mathrm{out}_k(x_{\ell+3},x_{\ell+4}))] \wedge$$
$$[\neg\mathrm{in}_k^{\lhd \ell+2}(x_1,\ldots,x_{\ell+4}) \to (\mathrm{in}_k^{\ell+2}(x_1,\ldots,x_{\ell+4}) \leftrightarrow \mathrm{out}_k(x_{\ell+3},x_{\ell+4}))].$$

Suppose that a, a' are k-integers, b, b' $(k{+}1)$-integers and $\bar{c}$ an ℓ-tuple of elements. From (Φ_{12}), $\mathfrak{A} \models \mathrm{in}_k^{\ell+4}[a,b,\bar{c},b',a'] \Leftrightarrow \mathfrak{A} \models \mathrm{in}_k[a,b]$, and from (Φ_{13}), $\mathfrak{A} \models \mathrm{in}_k^{\lhd \ell+2}[a,b,\bar{c},b',a'] \Leftrightarrow \mathfrak{A} \models \mathrm{in}_k^{\lhd}[a,b]$. Furthermore, by $(k{+}1)$-harmony, $\mathfrak{A} \models \mathrm{out}_k[b',a'] \Leftrightarrow \mathfrak{A} \models \mathrm{in}_k[a',b']$. Hence, from our preliminary remarks on the canonical representations of numbers by bit-strings, $\mathfrak{A} \models \varrho_{k,\ell}[a,b,\bar{c},b',a']$ just in case the $\mathrm{val}_k(a')$-th digit in the encoding of $\mathrm{val}_{k+1}(b')$ is the same as the $\mathrm{val}_k(a)$-th digit in the encoding of $\mathrm{val}_{k+1}(b) - 1$, modulo $\mathfrak{t}(k+1,n)$.

Now add to $\Sigma_{m,n}$ the ternary predicate $\mathrm{predDig}_{k+1,0}$ and quaternary predicate $\mathrm{predDig}_{k+1,1}$, and add to $\Phi_{m,n}$ the conjunct

$$\bigwedge_{\ell=0}^{1} \forall x_1(\mathrm{int}_k(x_1) \to \forall x_2(\mathrm{int}_{k+1}(x_2) \to \forall x_3 \cdots \forall x_{\ell+3}(\mathrm{int}_{k+1}(x_{\ell+3}) \to$$
$$\forall x_{\ell+4}(\mathrm{int}_k(x_{\ell+4}) \wedge \mathrm{eq}_{k,\ell+2}(x_1,\ldots,x_{\ell+4}) \to$$
$$(\mathrm{predDig}_{k+1,\ell}(x_2,\ldots,x_{\ell+4}) \leftrightarrow \varrho_{k,\ell}(x_1,\ldots,x_{\ell+4}))))). \qquad (\Phi_{14})$$

This formula is illustrated in the left-hand diagram of Fig. 2 in the case $\ell = 1$: here, $\varrho_{k,1}$ holds of the tuple a,b,c,b',a, and $\mathrm{predDig}_{k+1,1}$ of the tuple b,c,b',a, just in case the $\mathrm{val}_k(a)$th digit of $\mathrm{val}_{k+1}(b')$ agrees with the $\mathrm{val}_k(a)$th digit of $\mathrm{val}_{k+1}(b) - 1$. (Note that the single element a is depicted twice in this diagram.) Suppose a is a k-integer, b, b' are $(k+1)$-integers in $\mathfrak{A}$, and $\bar{c}$ is any ℓ-tuple from A with $0 \leq \ell \leq 1$. By k-equality (IH), $\mathfrak{A} \models \mathrm{eq}_{k,\ell+2}[a,b,\bar{c},b',a]$, and from the properties of $\varrho_{k,\ell}$ just established (setting $a' = a$), $\mathfrak{A} \models \mathrm{predDig}_{k+1,\ell}[b,\bar{c},b',a]$ just in case the $\mathrm{val}_k(a)$th digit of $\mathrm{val}_{k+1}(b')$ is equal to the $\mathrm{val}_k(a)$th digit of $\mathrm{val}_{k+1}(b) - 1$, modulo $\mathfrak{t}(k+1,n)$.

To establish $(k+1)$-predecessor, therefore, we add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{1} \forall x_1(\mathrm{int}_{k+1}(x_1) \to \forall x_2 \cdots \forall x_{\ell+2}(\mathrm{int}_{k+1}(x_{\ell+2}) \to$$
$$(\mathrm{pred}_{k+1,\ell}(x_1,\ldots,x_{\ell+2}) \leftrightarrow \forall x_{\ell+3}(\mathrm{int}_k(x_{\ell+3}) \to \mathrm{predDig}_{k+1,\ell}(x_1,\ldots,x_{\ell+3}))))). \quad (\Phi_{15})$$

From (Φ_{15}), $\mathfrak{A} \models \mathrm{pred}_{k+1,\ell}[b,\bar{c},b']$ just in case each digit of $\mathrm{val}_{k+1}(b')$ is equal to the corresponding digit of $\mathrm{val}_{k+1}(b) - 1$, modulo $\mathfrak{t}(k+1,n)$.

If, on the other hand, $k+1 = m$, we proceed as above, but we add to $\Sigma_{m,n}$ only the predicates in_k^2, $\mathrm{in}_k^{\lhd 2}$, $\mathrm{predDig}_{k+1,0}$ (not in_k^3, $\mathrm{in}_k^{\lhd 3}$ or $\mathrm{predDig}_{k+1,1}$), and we add to $\Phi_{m,n}$ only those conjuncts of (Φ_{12})–(Φ_{15}) with $\ell = 0$ (not with $\ell = 1$). This suffices for $(k+1)$-predecessor in the case $k+1 = m$, and does not require the use of any predicates outside $\Sigma_{m,n}$.

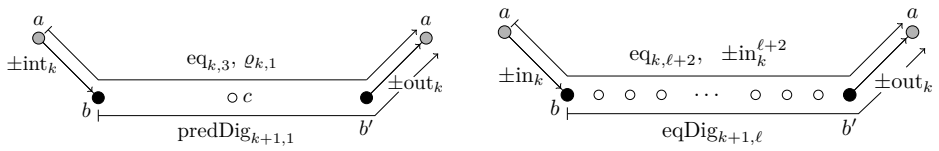


Figure 2 Fixing the interpretations of $\mathrm{predDig}_{k+1,1}$ (left) and $\mathrm{eqDig}_{k+1,\ell}$ (right).

To establish the property $(k+1)$-covering, we add to $\Phi_{m,n}$ the conjuncts

$$\exists x_1(\mathrm{int}_{k+1}(x_1) \wedge \mathrm{zero}_{k+1}(x_1)) \tag{Φ_{16}}$$

$$\forall x_1(\mathrm{int}_{k+1}(x_1) \to \exists x_2(\mathrm{int}_{k+1}(x_2) \wedge \mathrm{pred}_{k+1,0}(x_1, x_2))). \tag{Φ_{17}}$$

Note that (Φ_{17}) features only $\mathrm{pred}_{k+1,0}$, and not $\mathrm{pred}_{k+1,1}$, so it is defined even when $k+1=m$

It remains only to establish $(k+1)$-equality. Conceptually, this is rather easier than $(k+1)$-predecessor; however, we do need to consider larger numbers of semantically inert variables. Let $L = 2(m-k-1)$. The property $(k+1)$-equality concerns the interpretation of the $(\ell+2)$-ary predicate $\mathrm{eq}_{k+1,\ell}$ for for all ℓ $(0 \le \ell \le L)$. Observe that, if $k=1$ (first inductive step), then $L = 2m-4$, and if $k = m-1$ (last inductive step), then $L = 0$. Thus, in the sequel, we always have $L \le 2m-4$. (Remember that the inductive case is encountered only if $m \ge 2$.)

To ease the pain of reading, we split the task into three stages. For the first stage, for all ℓ $(0 \le \ell \le L)$, add to $\Sigma_{m,n}$ an $(\ell+2)$-ary predicate in_k^ℓ, and add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{L} \forall x_1(\mathrm{int}_k(x_1) \to \forall x_2(\mathrm{int}_{k+1}(x_2) \wedge \pm\mathrm{in}_k(x_1,x_2) \to \forall x_3 \cdots \forall x_{\ell+2} \pm \mathrm{in}_k^\ell(x_1, x_2, \ldots, x_{\ell+1}, x_{\ell+2}))), \tag{Φ_{18}}$$

thus fixing in_k^ℓ to be the result of adding ℓ semantically inert arguments to in_k^ℓ. (For $\ell \le 3$, this repeats the work of (Φ_{12}), but no matter.)

In the second stage, for all ℓ $(0 \le \ell \le L)$, add to $\Sigma_{m,n}$ an $(\ell+3)$-ary predicate $\mathrm{eqDig}_{k+1,\ell}$, and add to $\Phi_{m,n}$ the conjuncts

$$\bigwedge_{\ell=0}^{L} \forall x_1(\mathrm{int}_k(x_1) \to \forall x_2(\mathrm{int}_{k+1}(x_2) \to \forall x_3 \cdots \forall x_{\ell+3}(\mathrm{int}_{k+1}(x_{\ell+3}) \to \forall x_{\ell+4}(\mathrm{int}_k(x_{\ell+4}) \wedge \mathrm{eq}_{k,\ell+2}(x_1, \ldots, x_{\ell+4}) \to (\mathrm{eqDig}_{k+1,\ell}(x_2, \ldots, x_{\ell+4}) \leftrightarrow \eta_{k,\ell+2}(x_1, \ldots, x_{\ell+4})))))), \tag{Φ_{19}}$$

where $\eta_{k,\ell+2}(x_1, \ldots, x_{\ell+4})$ is the formula: $\mathrm{in}_k^{\ell+2}(x_1, \ldots, x_{\ell+4}) \leftrightarrow \mathrm{out}_k(x_{\ell+3}, x_{\ell+4})$.

Let b, b' be $(k+1)$-integers in $\mathfrak{A}$, a a k-integer in $\mathfrak{A}$, and $\bar{c}$ any ℓ-tuple from A. We claim that $\mathfrak{A} \models \mathrm{eqDig}_{k+1,\ell}[b, \bar{c}, b', a]$ just in case $\mathrm{val}_{k+1}(b)$ and $\mathrm{val}_{k+1}(b')$ agree on their $\mathrm{val}_k(a)$th bit. For, by k-equality (IH), $\mathfrak{A} \models \mathrm{eq}_{k,\ell+2}[a, b, \bar{c}, b', a]$. Hence, by (Φ_{19}) $\mathfrak{A} \models \mathrm{eqDig}_{k+1,\ell}[b, \bar{c}, b', a]$ holds just in case $\mathfrak{A} \models \eta_{k,\ell+2}[a, b, \bar{c}, b', a]$. But, by (Φ_{18}), $\mathfrak{A} \models \mathrm{in}_k^{\ell+2}[a, b, \bar{c}, b', a]$ if and only if $\mathfrak{A} \models \mathrm{in}_k[a, b]$, i.e. if and only if the $\mathrm{val}_k(a)$th bit of $\mathrm{val}_{k+1}(b)$ is 1. That is, $\mathfrak{A} \models \mathrm{eqDig}_{k+1,\ell}[b, \bar{c}, b', a]$ is equivalent to the statement that $\mathfrak{A} \models \mathrm{in}_k[a, b]$ if and only if $\mathfrak{A} \models \mathrm{out}_k[b', a]$. The situation is illustrated (for the case where $\mathfrak{A} \models \mathrm{eqDig}_{k+1,\ell}[b, \bar{c}, b', a]$ holds) in the right-hand diagram of Fig. 2, where all polarity alternatives $\pm$ are assumed to be resolved in the same way.

But, by $(k+1)$-harmony, $\mathfrak{A} \models \mathrm{out}_k[b', a]$ if and only if $\mathfrak{A} \models \mathrm{in}_k[a, b']$. This establishes the claim.

In the third stage, we add to $\Phi_{m,n}$ the conjunct

$$\bigwedge_{\ell=0}^{L} \forall x_1(\mathrm{int}_{k+1}(x_1) \to \forall x_2 \cdots \forall x_{\ell+2}(\mathrm{int}_{k+1}(x_{\ell+2}) \to (\mathrm{eq}_{k+1,\ell}(x_1,\ldots,x_{\ell+2}) \leftrightarrow \forall x_{\ell+3}(\mathrm{int}_k(x_{\ell+3}) \to \mathrm{eqDig}_{k+1,\ell}(x_1,\ldots,x_{\ell+3})))))). \qquad (\Phi_{20})$$

Given the properties of $\mathrm{eqDig}_{k+1,\ell}$ just established, this evidently secures $(k+1)$-equality, completing the induction.

We have remarked that, by m-covering, any model of $\Phi_{m,n}$ has cardinality at least $\mathfrak{t}(m,n)$. We claim that $\Phi_{m,n}$ is satisfiable. Let $A = A_1 \dot\cup \cdots \dot\cup A_m$, where $A_k = \{\langle k,i\rangle \mid 0 \le i < \mathfrak{t}(k,n)\}\rangle$. (That is, A is the disjoint union of the various sets of integers $[0,\mathfrak{t}(k,n)-1]$.) Let $\mathrm{int}_k^{\mathfrak{A}} = A_k$ for all k $(1 \le k \le m)$, and interpret the other predicates of $\Sigma_{m,n}$ as described. It is easily verified that $\mathfrak{A} \models \Phi_{m,n}$.

It remains only to check the number of variables featured in $\Phi_{m,n}$. Consider first the conjuncts introduced in the base case. By inspection, (Φ_1)–(Φ_3) and (Φ_5)–(Φ_6) are in $\mathcal{FL}^{2m}$. For $m > 1$, (Φ_4) is in $\mathcal{FL}^3$; but if $m = 1$, only the conjunct with index $\ell = 0$ is present, which is in $\mathcal{FL}^2$. Either way, (Φ_1)– (Φ_6) are in $\mathcal{FL}^{2m}$. Consider now the conjuncts introduced in the inductive case. By inspection, these feature only $\max(5,2m) \le 2m$ variables. If, however, $m = 2$, then the inductive step only runs once, with $k+1 = m$, in which case only those conjuncts of (Φ_{12})–(Φ_{15}) occur for which $\ell = 0$, which feature only 4 variables. Either way, (Φ_7)–(Φ_{20}) are in $\mathcal{FL}^{2m}$. ◀

Now that we can enforce m-tuply exponentially large models in $\mathcal{FL}^{2m}$, it is a simple matter to establish that the satisfiability problem for this fragment is m-NExpTime-hard. The technique involves the encoding of tiling problems over a grid of m-tuply exponential size using formulas of $\mathcal{FL}^{2m}$, the coordinates of the various positions in this grid being represented as pairs of m-integers. The details are roughly analogous to the NExpTime-hardness proof for the two-variable fragment of first-order logic (see e.g. [2], pp. 253 ff.), and are relegated to the Appendix.

▶ **Theorem 2.** *The satisfiability problem for $\mathcal{FL}^{2m}$ is m-NExpTime-hard. Hence, the satisfiability problem for $\mathcal{FL}$ is non-elementary.*

4 Upper bound

In this section we show that $\mathcal{FL}$ is decidable, thus confirming the Purdy's original article [8]. The strategy adopted here, however, employs a variant of a construction found in the (as we now know) flawed article [10]. As well as providing m-NExpTime upper complexity bounds for the sub-fragments $\mathcal{FL}^m$ for all $m \ge 1$, the argument below yields a much shorter and more perspicuous proof than that given in [8].

The 1-variable fluted fragment, $\mathcal{FL}^1$, coincides with the 1-variable fragment of first-order logic, and so its satisfiability (= finite satisfiability) problem is in NPTime (and hence certainly in NExpTime). Furthermore, the 2-variable fluted fragment, $\mathcal{FL}^2$, is a proper subset of the 2-variable fragment of first-order logic, whose satisfiability (= finite satisfiability) problem is known to be in NExpTime [3] (and hence certainly in 2-NExpTime). In the sequel, therefore, we may confine attention to the case $m \ge 3$.

We begin by narrowing the range of fluted formulas we need to consider. An $\mathcal{FL}^m$-sentence Φ is in *normal form* if it is a conjunction of sentences of the forms:

$$\forall x_1 \forall x_2 \ldots \forall x_k(\theta \to \exists x_{k+1}\xi) \qquad \forall x_1 \forall x_2 \ldots \forall x_k(\vartheta \to \forall x_{k+1}\psi),$$

where $1 \le k < m$, and θ, ξ, ϑ, ψ are quantifier-free fluted formulas such that $\theta, \vartheta \in \mathcal{FL}^{[k]}$, and $\xi, \psi \in \mathcal{FL}^{[k+1]}$. The proof of the following lemma is routine:

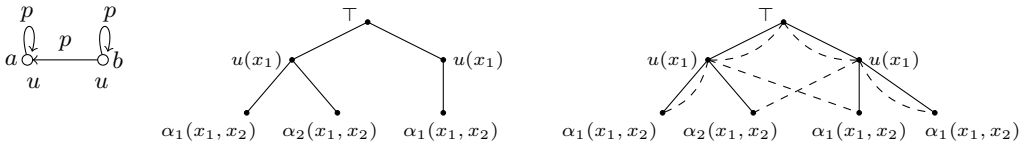


Figure 3 A structure $\mathfrak{A}$, its characteristic fluted (2,0)-constituent, $\mu = \mathrm{fc}_2^{\mathfrak{A}}[\,]$, depicted as a tree, and a double-tree formed from μ featuring a duplicated node.

▶ **Lemma 3.** *Let φ be an $\mathcal{FL}^m$-sentence over a signature Σ. We can compute, in exponential time, a disjunction $\Psi = \bigvee_{i\in I} \Psi_i$ of normal form $\mathcal{FL}^m$-sentences over a signature Σ' such that φ is satisfiable over a given domain A if and only if Ψ is satisfiable over A, $\|\Psi_i\| = O(\|\varphi\| \log\|\varphi\|)$ $(i \in I)$ and Σ' consists of Σ together with some additional predicates of arity at most $m-1$.*

We require one or two additional technical preliminaries. An *atomic fluted* k-type (over a given signature) is a maximal consistent conjunction of atomic or negated atomic $\mathcal{FL}^{[k]}$-formulas. For example, over a signature Σ featuring a single unary predicate u and a single binary predicate p, the formulas $\alpha_1(x_1, x_2) = u(x_2) \wedge p(x_1, x_2)$ and $\alpha_2(x_1, x_2) = u(x_2) \wedge \neg p(x_1, x_2)$ are atomic fluted 2-types. We take $\top$ to be the unique atomic fluted 0-type. Where the signature is clear from context, we denote the set of all atomic fluted k-types by $\boldsymbol{\alpha}^{(k)}$, and write $\boldsymbol{\alpha} = \bigcup_{k=0}^{m} \boldsymbol{\alpha}^{(k)}$. Observe that $|\boldsymbol{\alpha}^{(k)}|$ for each k is bounded by $2^{|\Sigma|}$, hence $|\boldsymbol{\alpha}| \leq m \cdot 2^{|\Sigma|}$. For a given Σ-structure $\mathfrak{A}$ and $\bar{a} \in A^k$, we denote by $\mathrm{tp}^{\mathfrak{A}}[\bar{a}]$ the unique atomic fluted k-type t such that $\mathfrak{A} \models t[\bar{a}]$.

We employ an adaptation, to the fluted case, of the familiar notion of Hintikka constituent (see, e.g. [14]). Again, fix some signature Σ. Define a *fluted* (m, m)*-constituent* to be an atomic fluted m-type. Let k satisfy $m > k \geq 0$. Define a *fluted* (k, m)*-constituent* to be a formula

$$\lambda = t(x_1, \ldots, x_k) \wedge \bigwedge_{\lambda' \in \Lambda} \exists x_{k+1}.\lambda' \wedge \forall x_{k+1} \bigvee_{\lambda' \in \Lambda} \lambda', \tag{6}$$

where Λ is some set of fluted $(k+1, m)$-constituents, and t an atomic fluted k-type. We call the elements of Λ the *successors* of λ, and t the *atomic part of* λ. Any fluted (k, m)-constituent λ has a natural representation as a labelled tree, with the children of each node given by its successors and the label by its atomic part.

We illustrate these notions for the signature Σ considered above, and for the value $m = 2$. The fluted $(2, 2)$-constituents are, by definition, the atomic fluted 2-types. Hence, the formulas $\lambda_1(x_1) = u(x_1) \wedge \exists x_2.\alpha_1(x_1, x_2) \wedge \exists x_2.\alpha_2(x_1, x_2) \wedge \forall x_2(\alpha_1(x_1, x_2) \vee \alpha_2(x_1, x_2))$ and $\lambda_2(x_1) = u(x_1) \wedge \exists x_2.\alpha_1(x_1, x_2) \wedge \forall x_2.\alpha_1(x_1, x_2)$ are fluted (1,2)-constituents. In the same way, the sentence $\mu = \exists x_1 \lambda_1(x_1) \wedge \exists x_1 \lambda_2(x_1) \wedge \forall x_1(\lambda_1(x_1) \vee \lambda_2(x_1))$ is a fluted (0,2)-constituent. The tree corresponding to this fluted (0,2)-constituent is depicted in the middle diagram of Fig. 3. By elementary combinatorics, one can show that the number of fluted (k, m)-constituents is bounded by $\mathfrak{t}(m - k + 1, n + m - k)$, where $n = |\Sigma|$.

The following lemma states that, for fixed k and m, the fluted (k, m)-constituents over a given signature form a partition. The proof proceeds in exactly the same way as for Hintikka constituents (cf. Theorem 2 in [8] and Theorem 3.10 in [14]).

▶ **Lemma 4.** *Fix some signature Σ and integers $m \geq k \geq 0$. Denote by Λ the set of fluted (k, m)-constituents over Σ. Then: (i) $\models \bigvee \Lambda$; and (ii) $\models \lambda \to \neg\mu$ for all distinct $\lambda, \mu \in \Lambda$.*

It follows that, in any structure $\mathfrak{A}$, and for any m, k ($m \geq k \geq 1$), a k-tuple $\bar{a}$ from A satisfies a unique fluted (k,m)-constituent. We denote this fluted (k,m)-constituent by $\mathrm{fc}_m^{\mathfrak{A}}[\bar{a}]$. In the case $k = 0$, we obtain the fluted $(0,m)$-constituent $\mathrm{fc}_m^{\mathfrak{A}}[\,]$, which we call the *characteristic* fluted $(0,m)$-constituent of $\mathfrak{A}$. The left-hand diagram in Fig. 3 shows a 2-element structure $\mathfrak{A}$ over domain $\{a,b\}$ whose characteristic fluted $(0,m)$-constituent is the sentence μ in our example above. Indeed, we see that $\mathrm{fc}_2^{\mathfrak{A}}[a] = \lambda_1$ and $\mathrm{fc}_2^{\mathfrak{A}}[b] = \lambda_2$.

Let λ be a fluted (k,m)-constituent. If $k > 0$, define $\varphi^{\leftarrow}$ to be the formula obtained from φ by deleting all literals of λ containing x_1 and then shifting remaining variables left, i.e. replacing each variable x_{i+1} by x_i, for $i > 1$. Thus, continuing the example of the previous paragraph, we have (removing unnecessarily duplicated conjuncts) $\lambda_1^{\leftarrow} = \exists x_1.u(x_1) \wedge \forall x_1.u(x_1)$. If, on the other hand, $k < m$, define $\lambda^{\uparrow}$ to be the formula obtained by removing from λ all literals containing x_m and all subformulas starting with a quantifier binding x_m. (If $k = m = 1$, we take $\lambda^{\leftarrow}$ to be $\top$; similarly, if $k = 0$ and $m = 1$, we take $\lambda^{\uparrow}$ to be $\top$.) The following Lemma is completely routine.

▶ **Lemma 5.** *Let λ be a fluted (k,m)-constituent. If $k > 0$, then $\lambda^{\leftarrow}$ is a fluted $(k-1, m-1)$-constituent; if, on the other hand, $k < m$, then $\lambda^{\uparrow}$ is a fluted $(k, m-1)$-constituent. Moreover, if $\lambda = \mathrm{fc}_m^{\mathfrak{A}}[\,]$ for some Σ-structure $\mathfrak{A}$ and λ' is a fluted $(1,m)$-constituent that is a successor of λ, then $\lambda'^{\leftarrow} = \lambda^{\uparrow}$.*

By way of motivation, suppose $\mathfrak{A}$ is a structure, with $\mathrm{fc}_m^{\mathfrak{A}}[a_1, \ldots, a_k] = \lambda$. If $k > 0$, then $\mathrm{fc}_{m-1}^{\mathfrak{A}}[a_2, \ldots, a_k] = \lambda^{\leftarrow}$; likewise, if $k < m$, then $\mathrm{fc}_{m-1}^{\mathfrak{A}}[a_1, \ldots, a_k] = \lambda^{\uparrow}$.

For the purposes of this section it is expedient to view a (finite, non-empty) tree as a tuple $T = \langle V, \varepsilon, p\rangle$, where $\varepsilon \in V$ and $p : (V \setminus \{\varepsilon\}) \to V$ is a function such that, for all $v \in V$, $p(\cdots p(v)) = \varepsilon$ for some finite (possibly zero) number of applications of p. The elements of V are the nodes of T, ε is the root, and, for every other node v, $p(v)$ is the parent of v. The number $h(v)$ of applications of p required to reach ε from v is the *height* of v; thus, $h(\varepsilon) = 0$. As usual, for a subset $X \subseteq V$, $p[X]$ denotes the image of X, and $p^{-1}[X]$, the inverse image of X under p. We write $p^{-1}[v]$ for $p^{-1}[\{v\}]$ – that is, the children of v. As usual, a *leaf* of a tree is a node v with no children, and we say that a tree is *uniform* if all leaves have the same height. The *height* of a uniform tree is the height of any of its leaves.

We now define the main notions for this section. Let $T_f = \langle V, \varepsilon, f\rangle$ and $T_g = \langle V, \varepsilon, g\rangle$ be uniform trees of height m sharing the same nodes and root. We say $\mathbb{T} = \langle V, \varepsilon, f, g\rangle$ is a *double tree* if the following hold:

$$\text{for every } v \in V,\ f^{-1}[\varepsilon] = g^{-1}[\varepsilon]; \tag{D1}$$

$$\text{for every } v \in V \setminus \{\varepsilon\}, \text{ if } f^{-1}[v] \neq \emptyset, \text{ then } g[f^{-1}[v]] = f^{-1}[g(v)]. \tag{D2}$$

A routine induction shows that, if $\mathbb{T} = \langle V, \varepsilon, f, g\rangle$ is a double tree, then, for all $v \in V$, v has the same height in the trees $\langle V, \varepsilon, f\rangle$ and $\langle V, \varepsilon, g\rangle$. We call this number the *height* of v in $\mathbb{T}$, and denote it by $h(v)$. Note that (D2) is, in essence, a *confluence condition*: if m is the height of $\mathbb{T}$, then, for any node v (with $1 \leq h(v) < m$) the children of v in T_f map under g onto the children of $u = g(v)$ in T_f (see Fig. 4).

A moment's thought shows that not all trees can be made into double trees. This is true, for example, of the tree depicted in the middle diagram of Fig. 3: the right-hand node labelled $u(x_1)$ has exactly one child and one proper sister, which makes it impossible to satisfy (D2). On the other hand, by duplicating nodes if necessary, double trees can be created, as shown in the right-hand diagram of Fig. 3, where the function g is indicated by dashed edges. This process of transforming trees (specifically, trees depicting characteristic fluted constituents of structures) into double trees plays a crucial role in the proof of Lemma 7, below.

Let $\mathbb{T} = \langle V, \varepsilon, f, g\rangle$ be a double tree and $\tau : V \to \boldsymbol{\alpha}$ a function. We say $\mathbb{T} = \langle V, \varepsilon, f, g, \tau\rangle$ is an $\boldsymbol{\alpha}$-*double tree* if the following conditions hold:

$$\text{for every } v \in V,\ \tau(v) \in \boldsymbol{\alpha}^{(k)}, \text{ where } k = h(v); \tag{D3}$$

$$\text{for every } v \in V \setminus \{\varepsilon\},\ \tau(g(v)) = \tau(v)^{\leftarrow}. \tag{D4}$$

Thus, in an $\boldsymbol{\alpha}$-*double tree*, every node v is labelled with some atomic fluted $h(v)$-type, α. Notice that, if $h(v) \geq 1$, then $g(v)$ must be labelled with $\alpha^{\leftarrow}$.

Before we actually construct any $\boldsymbol{\alpha}$-double trees, let us first see what we can do with them. Let $\varphi \in \mathcal{FL}^m$ be a sentence in normal form, and $\mathbb{T} = \langle V, \varepsilon, f, g, \tau\rangle$ be an $\boldsymbol{\alpha}$-double tree. We say $\mathbb{T}$ *satisfies* φ, and write $\mathbb{T} \models \varphi$, when the following conditions hold for all $k < m$ and for every $v \in V$ with $h(v) = k$: (i) for every conjunct of φ of the form $\forall x_1 \forall x_2 \dots \forall x_k(\theta \to \exists x_{k+1}\xi)$, if $\tau(v) \models \theta$ then there is a $v' \in f^{-1}[v]$ such that $\tau(v') \models \xi$; and (ii) for every conjunct of φ of the form $\forall x_1 \forall x_2 \dots \forall x_k(\vartheta \to \forall x_{k+1}\psi)$, if $\tau(v) \models \vartheta$ then for every $v' \in f^{-1}[v]$, $\tau(v') \models \psi$. Thus, any $\boldsymbol{\alpha}$-double tree of height m endows any normal form formula of $\mathcal{FL}^m$ with a truth-value in a natural way. Observe that, in this definition, a conjunct of φ with variables $x_1, \dots, x_{k+1}$ imposes constraints *only* on the labels of nodes with height k and $k+1$. However, conditions (D1)–(D4) on $\boldsymbol{\alpha}$-double trees ensure that these constraints affect adjacent pairs of nodes throughout $\mathbb{T}$.

There is a close – though subtle – relationship between $\boldsymbol{\alpha}$-double trees and models of normal-form $\mathcal{FL}$-formulas, encapsulated in the following two lemmas.

▶ **Lemma 6.** *Let $\varphi \in \mathcal{FL}^m$ be in normal form, and suppose $\mathbb{T}$ is an $\boldsymbol{\alpha}$-double tree of height m, s.t. $\mathbb{T} \models \varphi$. Then there is a model $\mathfrak{A} \models \varphi$, with $|A|$ equal to the number of leaves in $\mathbb{T}$.*

Proof. Let A be the set of leaves of $\mathbb{T}$. We assign to each node v of $\mathbb{T}$ a subset $A_v \subseteq A$ with the property that, for each non-leaf node v, the family $\{A_w \mid w \text{ a child of } v\}$ partitions A into non-empty subsets, making crucial use of the properties (D1) and (D2) of double trees. In this way, any k-tuple $a_1, \dots a_k$ from A is naturally associated with a node v of height k, namely, that node v with ancestors $\varepsilon, v_1, \dots, v_k = v$ where, for all i $(1 \leq i \leq k)$, $a_i \in A_{v_i}$. This assignment of sequences to nodes allows us to turn A into a structure $\mathfrak{A}$: if $a_1, \dots a_k$ is associated with the node v, we set $\mathrm{tp}^{\mathfrak{A}}_m[a_1, \dots, a_k]$ to be whatever atomic fluted k-type $\mathbb{T}$ labels v with. Using the properties (D3)–(D4), this assignment can be shown to be consistent, and to result in a structure satisfying φ. The details are given below.

Let $\mathbb{T} = \langle V, \varepsilon, f, g, \tau\rangle$ be a finite $\boldsymbol{\alpha}$-double tree satisfying φ. We decompose V as a union of disjoint subsets $V = \bigcup_{k=0}^{m} V^k$, where V^k $(0 \leq k \leq h)$ is the set of nodes of height k. Let A be the set of leaves of $\mathbb{T}$ – this will be the domain of the model $\mathfrak{A}$ we are going to construct. First we define two labelling functions $d_0, d : V \mapsto \mathcal{P}(A)$ satisfying the following properties for each k $(0 \leq k \leq m)$:

- **(i)** for every $w \in V_k$, $d_0(w) \neq \emptyset$ and $d_0(w) \subseteq d(w)$,
- **(ii)** if $0 < k$ then, for every $w \in V^k$, $d(w) \subseteq d(g(w))$,
- **(iii)** $d(\varepsilon) = A$, and if $0 < k$, then, for every $w \in V^k$, the family $\{d(w') \mid w' \in f^{-1}[f(w)]\}$ is a partition of A.

We call $d(w)$ the *local domain* of w and $d_0(w)$ the *initial* local domain of w. We remark that, in (iii), $f^{-1}[f(w)]$ is the set of (reflexive) siblings of w in T_f.

For each element $w \in V$, we define $d_0(w)$ to be the set of leaves in the subtree of T_g rooted at w. Thus for any w with $h(w) \geq 1$,

$$d_0(w) \subseteq d_0(g(w)). \tag{7}$$

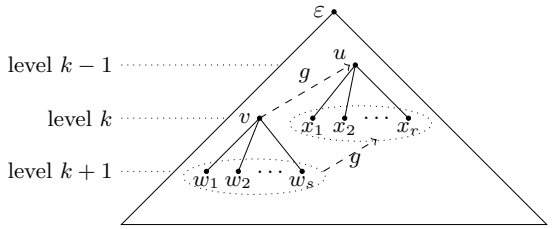


Figure 4 A double tree: $v \in V^k$, $w_1, \ldots, w_s$ are f-children of v; $x_1, \ldots, x_r$ are f-children of $u = g(v)$ and $g[\{w_1, \ldots, w_s\}] = \{x_1, \ldots, x_r\}$.

Now define $d(\varepsilon) = d_0(\varepsilon) = A$ and for each $w \in V^1$, define $d(w) = d_0(w)$. This ensures properties (i)–(iii) for $k = 0$ and $k = 1$. The function d is defined for remaining nodes by induction on the height. Assume properties (i)–(iii) hold for all nodes $v \in V^k$, where $1 \leq k < m$; we extend d to nodes $w \in V^{k+1}$ as follows. Consider first any element $v \in V^k$: we proceed to define $d(w)$ for every $w \in f^{-1}[v]$. Remembering that $k \geq 1$, let $u = g(v)$. Condition (D2) in the definition of double trees tells us that $g(f^{-1}[v]) = f^{-1}(u)$; we may therefore consider the various nodes in $x \in f^{-1}(u)$ one by one, defining $d(w)$ for all those $w \in f^{-1}[v]$ such that $g(w) = x$. Suppose $g^{-1}[x] = \{w_1, \ldots, w_l\}$. We define $d(w)$ for each of these nodes w by starting with the set $d_0(w)$, and then distributing the elements of $d(x) \setminus (d_0(w_1) \cup \ldots \cup d_0(w_l))$ among these sets in any way. Property (i) is thus secured trivially for all these w. From (7), we see that, for each such w, $d(w) \subseteq d(x) = d(g(w))$, thus securing (ii). By executing this procedure for all $x \in f^{-1}[u]$, we will have defined $d(w)$ for all $w \in f^{-1}[v]$. From property (iii) applied to level k, the family $\{d(x) \mid x \in f^{-1}[u]\}$ is a partition of A, whence the family $\{d(w) \mid w \in f^{-1}[v]\}$ will be a partition of A, thus securing property (iii) for all children of v in T_f. Therefore, after considering every $v \in V$ of height k, we obtain properties (i)–(iii) for all elements of V of height $k+1$.

We use one more piece of notation: for every $v \in V$ define S_v, the *sequence domain of* v, to be the set of k-tuples $S_v = d(v_1) \times \ldots \times d(v_k)$, where $\varepsilon, v_1, \ldots, v_k = v$ is the path in T_f from the root to v. Simple induction using property (iii) of the construction shows that sequence domains at each level k $(0 < k \leq m)$ form a partition of A^k:
(iv) $\bigcup_{v \in V^k} S_v = A^k$ and, if for some $\bar{a} \in A^k$ we have $\bar{a} \in S_v$ and $\bar{a} \in S_w$ then $v = w$.

Now we set the interpretation of the predicate letters on A. Namely, for every $p \in \Sigma$ of arity k, for every $v \in V$ with $h(v) \geq k$, and for every $\bar{a}\bar{b} \in S_v$ with $|\bar{b}| = k$, define:

$$\bar{b} \in p^{\mathfrak{A}} \quad \text{iff} \quad p(x_{l+1}, \ldots, x_{l+k}) \in \tau(v).$$

This is well defined. For, suppose in addition that $\bar{c}\bar{b} \in S_w$ for some $w \in V$ with $h(w) \geq k$. Let u and x be the ancestors of, respectively, v and w in T_g at level k. By property (ii) of the d-labelling, $\bar{b} \in S_u$ and $\bar{b} \in S_x$. Now, property (iv) implies $u = x$. And so applying repeatedly condition (D4) of the definition of an $\boldsymbol{\alpha}$-double tree to $\tau(v)$ and $\tau(w)$ we get:

$$p(x_{l+1}, \ldots, x_{l+k}) \in \tau(v) \quad \text{iff} \quad p(x_1, \ldots, x_k) \in \tau(u) \quad \text{iff} \quad p(x_{j+1}, \ldots, x_{j+k}) \in \tau(w).$$

It remains to check that $\mathfrak{A} \models \varphi$. Observe that for each $v \in V$ with $v \neq \varepsilon$ S_v is non-empty, and moreover for each $\bar{a} \in S_v$ $tp^{\mathfrak{A}}(\bar{a}) = \tau(v)$. Since φ is in normal form and $\mathbb{T}$ satisfies φ, it is obvious that all conjuncts of φ are true in $\mathfrak{A}$. ◀

▶ **Lemma 7.** *Let $\varphi \in \mathcal{FL}^m$ be in normal form, and suppose $\mathfrak{A} \models \varphi$. Then there exists a finite $\boldsymbol{\alpha}$-double tree satisfying φ with the number of leaves bounded by $\mathfrak{t}(m, O(mn))$.*

Proof. We begin by considering the characteristic fluted $(m,0)$-constituent $\mu = \mathrm{fc}_m^{\mathfrak{A}}[\,]$. As we have seen, μ can be viewed as a tree $T_\lambda = \langle V, f, \varepsilon\rangle$. Any node v at level k ($0 \leq k \leq m$) in T_λ corresponds to a position in the syntax tree of μ defining a fluted (k,m)-constituent v_λ. Note that we may have $\lambda_v = \lambda_w$ for distinct nodes v, w, because there may be repeated subformulas. Our task is to make T_λ into an $\boldsymbol{\alpha}$-double tree satisfying φ. We begin by adding a second parent function g to T_λ satisfying

$$\lambda^{\uparrow}_{g(v)} = \lambda^{\leftarrow}_v \text{ for all } v \in V \setminus \{\varepsilon\}, \tag{8}$$

which we now proceed to define.

We start by setting, for every v with height 1, $g(v) = f(v) = \varepsilon$. Note that Lemma 5 implies that for all nodes v with height 1, $\lambda^{\leftarrow}_v = \lambda^{\uparrow}_\varepsilon$, as required by (8). Now suppose that $g(v) = u$ has been defined for some node v of height $k < m$ in such a way that (8) holds, and suppose that w is a child of v, that is: $f(w) = v$. Thus, λ_w is simply a subformula of λ_v, whence, by (8), $\lambda_{g(v)}$ must have some subformula π such that $\pi^{\uparrow} = \lambda^{\leftarrow}_w$. Indeed, π is a fluted (k,m)-constituent. Let x be the child of u (i.e. $f(x) = u$) such that $\lambda_x = \pi$. Thus, $\lambda^{\leftarrow}_w = \lambda^{\uparrow}_x$, and we may set $g(w) = x$. Proceeding in this way, we can define g for all of $V \setminus \{\varepsilon\}$ satisfying (8). The resulting construction is illustrated in Fig. 4. Unfortunately, this construction does not *quite* ensure (D2); for, while g maps the children (under f) of v to the children (under f) of $g(v)$, the latter set may not be covered by this mapping. That is, we have $g[f^{-1}[v]] \subseteq f^{-1}[g(v)]$ rather than the desired $g[f^{-1}[v]] = f^{-1}[g(v)]$. However, this matter can be rectified by creating duplicates of the children of v which are then free to be mapped to any children of $g(v)$ not yet accounted for (c.f. Fig. 3).

To make this construction precise we need one more notion. Let k satisfy $0 < k < m$. A fluted (k,m)-*semi-constituent* is defined in the same way as a fluted (k,m)-constituent via the recursion in (6), except that Λ is now a finite *multiset* (rather than *set*) of fluted $(k+1,m)$-semi-constituents. In other words, fluted semi-constituents are just like fluted constituents, with the difference that repeated successors are allowed. Let η and ζ be fluted (k,m)-semi-constituents. We write $\eta \approx \zeta$ if, after removing repeated successors from η and ζ (at all levels in the recursion), the same fluted (k,m)-constituent is obtained. We represent fluted semi-constituents as trees, just as we do fluted constituents; moreover, the operations $^{\leftarrow}$ and $\uparrow$ on fluted constituents are extended to fluted semi-constituents in the obvious way.

Now, we are ready to give the details. To define the function g in T_λ, we start as mentioned above by setting, for every $w \in V^1$, $g(w) = f(w) = \varepsilon$. For nodes $w \in V$ with $h(w) > 1$, we define $g(w)$ assuming that $g(f(w))$ has already been defined and that the following condition holds for $v = f(w)$

$$\lambda^{\uparrow}_{g(v)} \approx \lambda^{\leftarrow}_v. \tag{9}$$

The basic step, for a single node w, is performed as follows.

> *Define* $g(w)$: suppose $v = f(w)$ and (9) holds for v. Since λ_w is a successor of λ_v there exists at least one conjunct η of $\lambda_{g(v)}$ such that $\eta^{\uparrow} \approx \lambda^{\leftarrow}_w$. So we may pick a node x such that $\lambda_x = \eta$ and we set $g(w) = x$.

Observe that after defining $g(w)$ as above we have $\lambda^{\uparrow}_{g(w)} \approx \lambda^{\leftarrow}_w$. So, we may extend the definition of g to a complete f-subtree $T_f(w)$ of w, traversing the subtree level by level. This is performed as follows.

> *Complete*$(g, T_f(w))$: suppose $v = f(w)$ and (9) holds for v.
> Let $z_1 \ldots, z_r$ be all the nodes of $T_f(w)$, excluding the root, ordered so that nodes

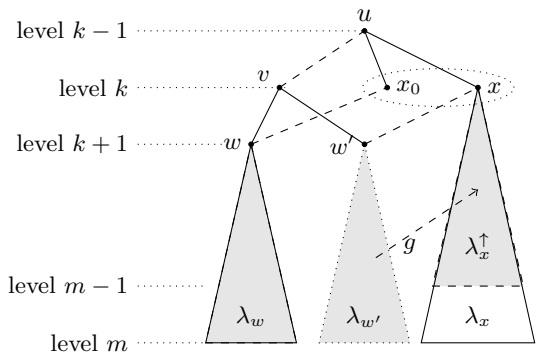


Figure 5 Covering x with respect to v: solid lines correspond to f and dashed lines – to g. $\lambda_w^{\leftarrow} \approx \lambda_x^{\uparrow}$ and $\lambda_{w'}$ is a copy of λ_w.

on lower levels appear before nodes on higher levels (e.g. according to breadth-first search). For every i $(0 \leq i \leq r)$ call *Define*$(g(z_i))$.

Now, we are ready to extend the definition of g to all nodes of V by calling the above procedure for every node $w \in V^1$. Denote the resulting tree by $T = \langle V, \varepsilon, f, g, \tau \rangle$. Evidently, in T the following properties hold for every k $(0 \leq k \leq m)$

(i) if $k > 0$, then for all $w \in V^k$, $\lambda_{g(w)}^{\uparrow} \approx \lambda_w^{\leftarrow}$,

(ii) if $0 < k < m$ then, for every $w \in V^k$, $g[f^{-1}[w]] \subseteq f^{-1}[g(w)]$.

In the required $\boldsymbol{\alpha}$-double tree to ensure (D2) the inclusions in condition (ii) are supposed to become equalities. This requires one more operation.

Suppose x and v are two nodes such that $g(v) = f(x)$ (refer to Figure 5). Suppose there is no node w such that $f(w) = v$ and $g(w) = x$. We then proceed as follows.

Covering x with respect to v: suppose $x \in V$, $1 \leq h(x) < m$, $g(v) = f(x) = u$, and $g^{-1}[x] = \emptyset$, where $h(x)$ is the height of x in the f-tree. By (i), $\lambda_u^{\uparrow} \approx \lambda_v^{\leftarrow}$. So, there is a conjunct η in λ_v such that $\eta^{\leftarrow} \approx \lambda_x^{\uparrow}$. Pick a node w such that $f(w) = v$ and $\lambda_w = \eta$. Add to v a copy $T_{w'}$ of the f-tree T_w. (We remind the reader that this is equivalent to add to λ_v a copy of the conjunct λ_w.) Define $g(w') = x$ and run the procedure *Complete*$(g, T_f(w'))$.

We note that the above operation maintains conditions (i)–(ii).

Now, we proceed on all nodes of the tree constructed, level by level, covering all the nodes x such that $h(x) < m$ with respect to all nodes v such that $g(v) = f(x)$. Denote the resulting tree by $\mathbb{T}$ and the set of nodes of $\mathbb{T}$ by $\mathbb{V}$.

Evidently, $\mathbb{T}$ is an $\boldsymbol{\alpha}$-double tree. Moreover, since we have added only identical conjuncts to T, $\mathbb{T}$ also satisfies φ.

It remains to estimate the number of leaves in $\mathbb{T}$. Let $C(k, m)$ be the number of fluted (k, m)-constituents and let $N(k)$ be the maximal number of children of a node $v \in \mathbb{V}$ with $h(v) = k$. By elementary combinatorics, one can show that $C(k, m) \leq \mathfrak{t}(m-k+1, n+m-k)$, where $n = |\Sigma|$. We have $N(0) = C(1, m)$, $N(1) \leq C(2, m) + C(1, m)$, and $N(m-1) \leq C(m, m) + \ldots + C(1, m) \leq m \cdot C(1, m)$. Hence, the number of leaves in $\mathbb{T}$ is bounded by $(m \cdot C(m-1))^m$, which is bounded by $\mathfrak{t}(m, O(mn))$. ◀

Lemmas 3, 6 and 7 instantly imply the main theorem of this section:

▶ **Theorem 8.** *$\mathcal{FL}$ has the finite model property. Moreover, the satisfiability problem for $\mathcal{FL}^m$ is in m-NExpTime.*

Discussion

When restricting attention to $\mathcal{FL}$ with a fixed number of variables, the upper complexity bounds given in Theorem 8 are not tight, even for $\mathcal{FL}^1$. As mentioned earlier, the 1-variable fluted fragment, $\mathcal{FL}^1$, coincides with the 1-variable fragment of first-order logic, and so its satisfiability problem is NPTime-complete (that is: 0-NExpTime-complete). Furthermore, the 2-variable fluted fragment, $\mathcal{FL}^2$, is a proper subset of the 2-variable fragment of first-order logic, whose satisfiability (= finite satisfiability) problem is known to be in NExpTime. Hence, setting $m = 1$ in Theorem 2, the satisfiability problem for $\mathcal{FL}^2$ is NExpTime-complete. By considering more closely the normal forms yielded by Lemma 3 and using a more complicated construction, the present authors have been able to strengthen Theorem 8 to show that, for $m \geq 3$, $\mathcal{FL}^m$ is in fact in $(m-2)$-NExpTime. Together with Theorem 2, this implies that $\mathcal{FL}^3$ is NExpTime-complete and $\mathcal{FL}^4$ is 2-NExpTime-complete. However, the additional gain in the upper complexity-bound is purchased at the cost of a less perspicuous proof, and anyway fails to close the gap with the lower complexity-bounds for $\mathcal{FL}^m$ when $m \geq 5$.

References

1. H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
2. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.
3. 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.
4. U. Hustadt, R. Schmidt, and L. Georgieva. A survey of decidable first-order fragments and description logics. *Journal of Relational Methods in Computer Science*, 1(3):251–276, 2004.
5. C. Lutz and U. Sattler. The complexity of reasoning with Boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 3, pages 1–20, Menlo Park, 2001. CLSI Publications.
6. A. Noah. Predicate-functors and the limits of decidability in logic. *Notre Dame Journal of Formal Logic*, 21(4):701–707, 1980.
7. W. C. Purdy. Decidability of fluted logic with indentity. *Notre Dame Journal of Formal Logic*, 37(1):84–104, 1996.
8. W. C. Purdy. Fluted formulas and the limits of decidability. *Journal of Symbolic Logic*, 61(2):608–620, 1996.
9. W. C. Purdy. Quine's limits of decision. *Journal of Symbolic Logic*, 64(4):1439–1466, 1999.
10. W. C. Purdy. Complexity and nicety of fluted logic. *Studia Logica*, 71:177–198, 2002.
11. W. V. Quine. On the limits of decision. In *Proceedings of the 14th International Congress of Philosophy*, volume III, pages 57–62. University of Vienna, 1969.
12. W. V. Quine. Algebraic logic and predicate functors. In *The Ways of Paradox*, pages 283–307. Harvard University Press, revised and enlarged edition, 1976.
13. W. V. Quine. The variable. In *The Ways of Paradox*, pages 272–282. Harvard University Press, revised and enlarged edition, 1976.
14. V. Rantala. Constituents. In R. Bogdan, editor, *Jaakko Hintikka*, volume 8 of *Profiles*, pages 43–76. Springer Netherlands, 1987.
15. R. Schmidt and U. Hustadt. A resolution decision procedure for fluted logic. In D. McAllester, editor, *Proceedings, CADE*, number 1831 in LNAI, pages 433–448. Springer-Verlag, 2000.

5 Appendix

Proof of Theorem 2. If $m = 1$, the result may be obtained by simple adaptation of the familiar proof that the satisfiability problem for the two-variable (non-fluted) fragment of first-order logic is NExpTime-hard [2, pp. 253, ff.]. We therefore assume in the sequel that $m \geq 2$. This avoids a tedious special case.

We employ the apparatus of tiling systems. A *tiling system* is a triple $\langle C, H, V\rangle$, where C is a non-empty, finite set and H, V are binary relations on C. The elements of C are referred to as *colours*, and the relations H and V as the *horizontal* and *vertical* constraints, respectively. For any integer N, a *tiling* for $\langle C, H, V\rangle$ of size N is a function $f : \{0, \ldots, N-1\}^2 \to C$ such that, for all i, j $(0 \leq i, j < N)$, the pair $\langle f(i,j), f(i+1,j)\rangle$ is in H and the pair $\langle f(i,j), f(i,j+1)\rangle$ is in V, with addition in arguments taken to be modulo N. A tiling of size N is to be pictured as a colouring of an $N \times N$ toroidal grid by the colours in C; the horizontal constraints H thus specify which colours may appear 'to the right of' which other colours; the vertical constraints V likewise specify which colours may appear 'above' which other colours. An n-tuple $\bar{c}$ of elements of C is an *initial configuration* for the tiling f if $\bar{c} = f(0,0), \ldots, f(n-1,0)$. An initial configuration for f is to be pictured as a row of n colours occupying the 'bottom left-hand' corner of the grid.

The *m-tuply exponential tiling problem* for a tiling system (C, H, V) is the following problem: given an n-tuple $\bar{c}$ from C, determine whether there exists a tiling for (C, H, V) of size $\mathfrak{t}(m, n)$ with initial configuration $\bar{c}$. Because of the close connection between runs of Turing machines and solutions of tiling systems, it is straightforward to see that there exist tiling systems for which the m-tuply exponential tiling problem is m-NExpTime-complete.

Let (C, H, V) be a tiling system and $m \geq 2$. We construct, for any n-tuple $\bar{c}$ from C, a formula $\Phi_{\bar{c}}$ over a signature $\Sigma_{\bar{c}}$, with the property that there exists a tiling for (C, H, V) of size $\mathfrak{t}(m, n)$ with initial configuration $\bar{c}$ if and only if $\Phi_{\bar{c}}$ is satisfiable. For ease of reading, we add predicates to $\Sigma_{\bar{c}}$ as they are encountered, and we specify the conjuncts of $\Phi_{\bar{c}}$ as and when they are required in the course of the reduction.

We begin by setting $\Sigma_{\bar{c}}$ to be the signature $\Sigma_{m,n}$ from Theorem 1, and we add to $\Phi_{\bar{c}}$ all the conjuncts of $\Phi_{m,n}$. In fact, it is the predicates established in the *penultimate* stage of the induction that interest us here. Specifically, in any model of $\Phi_{\bar{c}}$, we have a valuation function val_{m-1} defined on $(m-1)$-integers with values in the range $[1, \mathfrak{t}(m-1, n)]$, and satisfying the properties $(m-1)$-harmony, $(m-1)$-zero, $(m-1)$-predecessor, $(m-1)$-covering and $(m-1)$-equality. Using these predicates, we construct objects that may be thought of as, in essence, *pairs* of m-integers.

We start by adding to $\Sigma_{\bar{c}}$ the unary predicate vtx and the binary predicates in_X and in_Y. If a structure $\mathfrak{A}$ is clear from context, we call any element of A satisfying vtx a *vertex*. Define the function $\mathrm{val}_X : \mathrm{vtx}^{\mathfrak{A}} \to [0, \mathfrak{t}(m, n) - 1]$ by setting $\mathrm{val}_X(b)$, for any vertex b, to be the integer with canonical representation $s_{N-1}, \ldots, s_0$ of length $N = \mathfrak{t}(m-1, n)$ where, for all i $(0 \leq i < N)$,

$$s_i = \begin{cases} 1 \text{ if } \mathfrak{A} \models \mathrm{in}_X[a, b] \text{ for some } (m-1)\text{-integer } a \text{ such that} \\ \quad \mathrm{val}_{m-1}(a) = i; \\ 0 \text{ otherwise.} \end{cases}$$

Think of $\mathrm{val}_X(b)$ as the *horizontal coordinate* of b. The *vertical coordinate* of b, $\mathrm{val}_Y(b)$ is defined similarly, using the binary predicates in_Y. We rely on $(m-1)$-covering to ensure that, for any i in the range $[0, \mathfrak{t}(m-1, n) - 1]$, there *is* an $(m-1)$-integer having any value

i; and below we establish harmony-like properties showing that it does not matter *which* such $(m-1)$-integer we choose, if there are many.

Add to $\Sigma_{\bar{c}}$ the binary predicates out_X, out_Y, eq_X, eq_Y pred_X and pred_Y. Using D to stand for either of the letters X or Y, add to $\Phi_{\bar{c}}$ conjuncts ensuring the following properties

D-harmony: For all vertices b and all $(m-1)$-integers a, a' in $\mathfrak{A}$ such that $\mathrm{val}_{m-1}(a) = \mathrm{val}_{m-1}(a')$, $\mathfrak{A} \models \mathrm{in}_D[a,b] \Leftrightarrow \mathfrak{A} \models \mathrm{out}_D[a',b]$.

D-zero: For all vertices b in $\mathfrak{A}$, $\mathfrak{A} \models \mathrm{zero}_D[a] \Leftrightarrow \mathrm{val}_D(b) = 0$.

D-equality: For all vertices b, b' in $\mathfrak{A}$, $\mathfrak{A} \models \mathrm{eq}_D[b,b'] \Leftrightarrow \mathrm{val}_D(b) = \mathrm{val}_D(b')$.

D-predecessor: For all vertices b, b' in $\mathfrak{A}$, $\mathfrak{A} \models \mathrm{pred}_D[b,b'] \Leftrightarrow \mathrm{val}_D(b') = \mathrm{val}_D(b) - 1$ modulo $\mathfrak{t}(m,n)$.

The conjuncts in question are trivial adaptations of those used in the proof of Theorem 1 to establish m-harmony m-zero, m-equality and m-predecessor. All require at most $2m$ variables.

The following conjuncts of $\Phi_{\bar{c}}$ now establish that, for all pairs of integers i, j in the range $[0, \mathfrak{t}(m,n)-1]$, there exists a vertex a with coordinates (i,j):

$$\exists x_1(\mathrm{vtx}(x_1) \wedge \mathrm{zero}_X(x_1) \wedge \mathrm{zero}_Y(x_1))$$
$$\forall x_1(\mathrm{vtx}(x_1) \rightarrow \exists x_2(\mathrm{vtx}(x_2) \wedge \mathrm{pred}_Y(x_1,x_2) \wedge \mathrm{eq}_X(x_1,x_2)))$$
$$\forall x_1(\mathrm{vtx}(x_1) \rightarrow \exists x_2(\mathrm{vtx}(x_2) \wedge \mathrm{pred}_X(x_1,x_2) \wedge \mathrm{eq}_Y(x_1,x_2))).$$

Note that there is no requirement that vertices be uniquely defined by their horizontal and vertical coordinates.

Treating each colour in C as a unary predicate in $\Sigma_{\bar{c}}$, we colour each vertex uniquely by adding to $\Phi_{\bar{c}}$ the conjunct

$$\forall x_1(\mathrm{vtx}(x_1) \rightarrow \left(\bigvee_{c \in C} c(x_1)\right) \wedge \bigwedge_{c,d \in C}^{c \neq d} \neg(c(x_1) \wedge d(x_1))).$$

To obtain a well-defined grid-colouring, we wish models of $\Phi_{\bar{c}}$ to satisfy the following property.

chromatic harmony: For all vertices b and b' such that $\mathrm{val}_X(b) = \mathrm{val}_X(b')$ and $\mathrm{val}_Y(b) = \mathrm{val}_Y(b')$, and for all colours $c \in C$, $\mathfrak{A} \models c[b] \Leftrightarrow \mathfrak{A} \models c[b']$.

And this we ensure by adding to $\Phi_{\bar{c}}$ the conjunct

$$\forall x_1(\mathrm{vtx}(x_1) \wedge c(x_1) \rightarrow \forall x_2(\mathrm{vtx}(x_2) \wedge \mathrm{eq}_X(x_1,x_2) \wedge \mathrm{eq}_Y(x_1,x_2) \rightarrow c(x_2))).$$

Thus, any model of $\Phi_{\bar{c}}$ defines a colouring of the $\mathfrak{t}(m,n) \times \mathfrak{t}(m,n)$ toroidal grid. To ensure that this colouring is a tiling for (C,H,V), we simply add to $\Phi_{\bar{c}}$ the conjuncts

$$\bigwedge_{(c,d) \notin H} \forall x_1(\mathrm{vtx}(x_1) \wedge d(x_1) \rightarrow \forall x_2(\mathrm{vtx}(x_2) \wedge \mathrm{pred}_X(x_1,x_2) \wedge \mathrm{eq}_Y(x_1,x_2) \rightarrow \neg c(x_2)))$$
$$\bigwedge_{(c,d) \notin V} \forall x_1(\mathrm{vtx}(x_1) \wedge d(x_1) \rightarrow \forall x_2(\mathrm{vtx}(x_2) \wedge \mathrm{pred}_Y(x_1,x_2) \wedge \mathrm{eq}_X(x_1,x_2) \rightarrow \neg c(x_2))).$$

Finally, we need to ensure that $\bar{c}$ is written in the bottom left configuration. But this is routine. If c_0 is the initial element of $\bar{c}$, we write

$$\exists x_1(\mathrm{vtx}(x_1) \wedge \mathrm{zero}_X(x_1) \wedge \mathrm{zero}_Y(x_1) \wedge c_0(x_1)).$$

thus setting the 'bottom left' tile to have colour c_0. The remaining elements of $\bar{c}$ may be easily set using the predicates $\mathrm{predDig}_X$ and eq_Y. This completes the construction of $\Phi_{\bar{c}}$.

We have shown that, if $\Phi_{\bar{c}}$ is satisfiable, then the $\mathfrak{t}(m,n) \times \mathfrak{t}(m,n)$-grid colouring problem (C, H, V) has a solution. Conversely, a simple check shows that, if $\mathfrak{t}(m,n) \times \mathfrak{t}(m,n)$-grid colouring problem (C, H, V) has a solution, then by interpreting the predicates involved in $\Phi_{\bar{c}}$ as suggested above, we obtain a model of $\Phi_{\bar{c}}$. This completes the reduction. It is evident that, for fixed (C, H, V) and m, this reduction runs in time polynomially bounded by the length of $\bar{c}$. ◀

Free-Cut Elimination in Linear Logic and an Application to a Feasible Arithmetic*

Patrick Baillot[1] and Anupam Das[2]

1 **Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France**
2 **Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France**

Abstract

We prove a general form of 'free-cut elimination' for first-order theories in linear logic, yielding normal forms of proofs where cuts are anchored to nonlogical steps. To demonstrate the usefulness of this result, we consider a version of arithmetic in linear logic, based on a previous axiomatisation by Bellantoni and Hofmann. We prove a witnessing theorem for a fragment of this arithmetic via the 'witness function method', showing that the provably convergent functions are precisely the polynomial-time functions. The programs extracted are implemented in the framework of 'safe' recursive functions, due to Bellantoni and Cook, where the ! modality of linear logic corresponds to normal inputs of a safe recursive program.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases proof theory, linear logic, bounded arithmetic, polynomial time computation, implicit computational complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.40

1 Introduction

Free-cut elimination[1] is a normalisation procedure on formal proofs in systems including nonlogical rules, e.g. the axioms and induction rules in arithmetic, introduced in [26]. It yields proofs in a form where, essentially, each cut step has at least one of its cut formulas principal for a nonlogical step. It is an important tool for proving witnessing theorems in first-order theories, and in particular it has been extensively used in *bounded arithmetic* for proving complexity bounds on representable functions, by way of the *witness function method* [9].

Linear logic [14] is a decomposition of both intuitionistic and classical logic, based on a careful analysis of duplication and erasure of formulas. It has been useful in proofs-as-programs correspondences, proof search [1] and logic programming [24]. By controlling structural rules with designated modalities, the *exponentials*, linear logic has allowed for a fine study of complexity bounds in the Curry-Howard interpretation, inducing variants with polynomial-time complexity [17] [16] [18].

In this work we explore how the finer granularity of linear logic can be used to control complexity in *first-order theories*, restricting the provably convergent functions rather than the typable terms as in the propositional setting. We believe this to be of general interest,

* This work was supported by by the ANR Project ELICA ANR-14-CE25-0005 and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX- 0007) operated by the French National Research Agency (ANR).

[1] Also known as *anchored* or *directed* completeness, *partial* cut-elimination or *weak* cut-elimination in other works.

© Patrick Baillot and Anupam Das;
licensed under Creative Commons License CC-BY

in particular to understand the effect of substructural restrictions on nonlogical rules, e.g. induction, in mathematical theories. Some related works exist, e.g. the naïve set theories of Girard and Terui [15] [27], but overall it seems that the first-order proof theory of linear logic is still rather undeveloped; in particular, to our knowledge, there seems to be no general form of free-cut elimination available in the literature (although special cases occur in [22] and [3]). Thus our first contribution, in Sect. 3, is to provide general sufficient conditions on nonlogical rules for a first-order linear logic system to admit free-cut elimination.

We illustrate the usefulness of this result by proving a witnessing theorem for an arithmetic in linear logic, showing that the provably convergent functions are precisely the polynomial-time computable functions (Sects. 6 and 7), henceforth denoted **FP**. Our starting point is an axiomatisation $\mathcal{A}_2^1$ from [7], based on a modal logic, already known to characterise **FP**. This approach, and that of [20] before, differs from the bounded arithmetic approach since it does not employ bounds on quantifiers, but rather restricts nonlogical rules by substructural features of the modality [7] or by *ramification* of formulas [21]. The proof technique employed in both cases is a realisability argument, for which [20] operates directly in intuitionistic logic, whereas [7] obtains a result for a classical logic via a double-negation translation, relying on a higher-type generalisation of *safe recursion* [6].

We show that Buss' witness function method can be employed to extract functions directly for classical systems similar to $\mathcal{A}_2^1$ based in linear logic, by taking advantage of free-cut elimination. The De Morgan normal form available in classical (linear) logic means that the functions we extract remain at ground type, based on the usual safe recursive programs of [6]. A similar proof method was used by Cantini in [11], who uses combinatory terms as the model of computation as opposed to the equational specifications in this work.[2]

Our result holds for an apparently weaker theory than $\mathcal{A}_2^1$, with induction restricted to positive existential formulas in a way similar to Leivant's RT_0 system in [21] (see also [23]), but the precise relationship between the two logical settings is unclear. We conclude in Sect. 8 with a survey of related work and some avenues for further applications of the free-cut elimination result.

A version of this article containing further proof details in appendices is available [4].

2 Preliminaries

We formulate linear logic without units with usual notation for the multiplicatives, additives and exponentials from [14]. We restrict negation to the atoms, so that formulae are always in De Morgan normal form, and we also consider rules for arbitrary weakening when working in affine settings.

[2] This turns out to be important due to the handling of right-contraction steps in the witnessing argument.

▶ **Definition 1.** The sequent calculus for (affine) linear logic is as follows:[3]

$$\bot\text{-}l\,\frac{}{p, p^\bot \vdash} \qquad id\,\frac{}{p \vdash p} \qquad \bot\text{-}r\,\frac{}{\vdash p, p^\bot} \qquad cut\,\frac{\Gamma \vdash \Delta, A \quad \Sigma, A \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}$$

$$\parr\text{-}l\,\frac{\Gamma, A \vdash \Delta \quad \Sigma, B \vdash \Pi}{\Gamma, \Sigma, A \parr B \vdash \Delta, \Pi} \qquad \otimes\text{-}l\,\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \otimes B \vdash \Delta} \qquad \parr\text{-}r\,\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \parr B} \qquad \otimes\text{-}r\,\frac{\Gamma \vdash \Delta, A \quad \Sigma \vdash \Pi, B}{\Gamma, \Sigma \vdash \Delta, \Pi, A \otimes B}$$

$$\oplus\text{-}l\,\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \oplus B \vdash \Delta} \qquad \&\text{-}l\,\frac{\Gamma, A_i \vdash \Delta}{\Gamma, A_1 \& A_2 \vdash \Delta} \qquad \oplus\text{-}r\,\frac{\Gamma \vdash \Delta, A_i}{\Gamma \vdash \Delta, A_1 \oplus A_2} \qquad \&\text{-}r\,\frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \& B}$$

$$?\text{-}l\,\frac{!\Gamma, A \vdash ?\Delta}{!\Gamma, ?A \vdash ?\Delta} \qquad !\text{-}l\,\frac{\Gamma, A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad ?\text{-}r\,\frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, ?A} \qquad !\text{-}r\,\frac{!\Gamma \vdash ?\Delta, A}{!\Gamma \vdash ?\Delta, !A}$$

$$wk\text{-}l\,\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad cntr\text{-}l\,\frac{\Gamma, !A, !A \vdash \Delta}{\Gamma, !A \vdash \Delta} \qquad wk\text{-}r\,\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} \qquad cntr\text{-}r\,\frac{\Gamma \vdash \Delta, ?A, ?A}{\Gamma \vdash \Delta, ?A}$$

$$\exists\text{-}l\,\frac{\Gamma, A(a) \vdash \Delta}{\Gamma, \exists x.A(x) \vdash \Delta} \qquad \forall\text{-}l\,\frac{\Gamma, A(t) \vdash \Delta}{\Gamma, \forall x.A(x) \vdash \Delta} \qquad \exists\text{-}r\,\frac{\Gamma \vdash \Delta, A(t)}{\Gamma \vdash \Delta, \exists x.A(x)} \qquad \forall\text{-}r\,\frac{\Gamma \vdash \Delta, A(a)}{\Gamma \vdash \Delta, \forall x.A(x)}$$

where p is atomic, $i \in \{1, 2\}$, t is a term and the eigenvariable a does not occur free in Γ or Δ.

We do not formally include a symbol for implication but we sometimes write $A \multimap B$ as shorthand for $A^\bot \parr B$, where $A^\bot$ is the De Morgan dual of A. We often omit brackets under associativity, and when writing long implications we assume the right-most bracketing.

We will use standard terminology to track formulae in proofs, as presented in e.g. [10]. In particular, each rule has a distinguished *principal formula*, e.g. $A \parr B$ in the rule $\parr$-l (and similarly for all rules for the binary connectives) and $?A$ in the rule *cntr*-r, and *active formulae*, e.g. A and B in $\parr$-l and so on. These induce the notions of (direct) descendants and ancestors in proofs, as in [10].

2.1 Theories and systems

A *language* is a set of nonlogical symbols (i.e. constants, functions, predicates) and a *theory* is a set of closed formulae over some language. We assume that all theories contain the axioms of equality:

$$\begin{array}{l} \forall x.x = x \quad , \quad \forall x, y.(x = y \multimap y = x) \quad , \quad \forall x, y, z.(x = y \multimap y = z \multimap x = z) \\ \forall \vec{x}, \vec{y}.(\vec{x} = \vec{y} \multimap f(\vec{x}) = f(\vec{y})) \quad , \quad \forall \vec{x}, \vec{y}.(\vec{x} = \vec{y} \multimap P(\vec{x}) \multimap P(\vec{y})) \end{array} \tag{1}$$

where $\vec{x} = \vec{y}$ is shorthand for $x_1 = y_1 \otimes \cdots \otimes x_n = y_n$.

We consider *systems* of 'nonlogical' rules extending Dfn. 1, which we write as follows,

$$init\,\frac{}{\vdash A} \qquad (R)\,\frac{\{!\Gamma, \Sigma_i \vdash \Delta_i, ?\Pi\}_{i \in I}}{!\Gamma, \Sigma' \vdash \Delta', ?\Pi}$$

where, in each rule (R), I is a finite possibly empty set (indicating the number of premises) and we assume the following conditions and terminology:

[3] We consider a two-sided system since it is more intuitive for certain nonlogical rules, e.g. induction, and also convenient for the witness function method we use in Sect. 7.

1. In (R) the formulas of Σ', Δ' are called *principal*, those of Σ_i, Δ_i are called *active*, and those of $!\Gamma, ?\Pi$ are called *context formulas*. In *init* A is called a principal formula.
2. Each rule (R) comes with a list $a_1, \ldots, a_k$ of eigenvariables such that each a_j appears in exactly one Σ_i, Δ_i (so in some active formulas of exactly one premise) and does not appear in Σ', Δ' or $!\Gamma, ?\Pi$.
3. A system $\mathcal{S}$ of rules must be closed under substitutions of free variables by terms (where these substitutions do not contain the eigenvariables a_j in their domain or codomain).
4. In (R) the sequent Σ' (resp. Δ') does not contain any formula of the shape $?B$ (resp. $!B$), and in *init* the formula A is not of the form $!B$.

Conditions 2 and 3 are standard requirements for nonlogical rules, independently of the logical setting, cf. [5]. Condition 2 reflects the intuitive idea that, in our nonlogical rules, we often need a notion of *bound* variables in the active formulas (typically for induction rules), for which we rely on eigenvariables. Condition 3 is needed for our proof system to admit elimination of cuts on quantified formulas. Condition 4 is peculiar to our linear logic setting in order to carry out certain proof-theoretic manipulations for the free-cut elimination argument in Sect. 3.

Observe that *init* rules can actually be seen as particular cases of (R) rules, with no premise, so in the following we will only consider (R) rules.

To each theory $\mathcal{T}$ we formally associate the system of *init* rules $\vdash A$ for each $A \in \mathcal{T}$.[4] A proof in such a system will be called a *$\mathcal{T}$-proof*, or just proof when there is no risk of confusion.

▶ Remark (Semantics). The models we consider are usual Henkin models, with linear connectives interpreted by their classical counterparts. Consequently, we do not have any completeness theorem for our theories, but we do have soundness.

2.2 Some basic proof-theoretic results

We briefly survey some well-known results for theories of linear logic.

A rule is *invertible* if each of its upper sequents is derivable from its lower sequent.

▶ **Proposition 2** (Invertible rules, folklore). *The rules $\otimes$-l, $\parr$-r, $\oplus$-l, $\&$-r, $\exists$-l, $\forall$-r are invertible.*

We will typically write c-inv to denote the inverse derivation for a logical symbol c.

We also rely on the following result, which is also folklore but appeared before in [2].

▶ **Theorem 3** (Deduction, folklore). *For any theory $\mathcal{T}$ and closed formula A, $\mathcal{T} \cup \{A\}$ proves B if and only if $\mathcal{T}$ proves $!A \multimap B$.*

Due to these results notice that, in place of the equality axioms, we can work in a quantifier-free system of rules:

▶ **Proposition 4** (Equality rules). (1) *is equivalent to the following system of rules,*

$$\frac{}{\vdash t = t} \quad \frac{}{s = t \vdash t = s} \quad \frac{}{r = s, s = t \vdash r = t} \quad \frac{}{\vec{s} = \vec{t} \vdash f(\vec{s}) = f(\vec{t})} \quad \frac{}{\vec{s} = \vec{t}, P(\vec{s}) \vdash P(\vec{t})}$$

where r, s, t range over terms.

[4] Notice that this naively satisfies condition 3 since theories consist of only closed formulae.

3 Free-cut elimination in linear logic

We first define which cut instances may remain in proofs after free-cut elimination.

Since our nonlogical rules may have many principal formulae on which cuts may be anchored, we need a slightly more general notion of principality.

▶ **Definition 5.** We define the notions of *hereditarily principal formula* and *anchored cut* in a $\mathcal{S}$-proof, for a system $\mathcal{S}$, by mutual induction as follows:

- A formula A in a sequent $\Gamma \vdash \Delta$ is *hereditarily principal* for a rule instance (S) if either (i) the sequent is in the conclusion of (S) and A is principal in it, or (ii) the sequent is in the conclusion of an anchored cut, the direct ancestor of A in the corresponding premise is hereditarily principal for the rule instance (S), and the rule (S) is nonlogical.
- A cut-step is an *anchored cut* if the two occurrences of its cut-formula A in each premise are hereditarily principal for nonlogical steps, or one is hereditarily principal for a nonlogical step and the other one is principal for a logical step.

A cut which is not anchored will also be called a *free-cut*.

As a consequence of this definition, an anchored cut on a formula A has the following properties:

- At least one of the two premises of the cut has above it a sub-branch of the proof which starts (top-down) with a nonlogical step (R) with A as one of its principal formulas, and then a sequence of anchored cuts in which A is part of the context.
- The other premise is either of the same form or is a logical step with principal formula A.

Due to condition 4 in Sect. 2, we have the following:

▶ **Lemma 6.** *A formula occurrence A on the LHS (resp. RHS) of a sequent and hereditarily principal for a nonlogical rule (R) cannot be of the form $A = ?A'$ (resp. $A = !A'$).*

Now we can state the main result of this section:

▶ **Theorem 7** (Free-cut elimination)**.** *Given a system $\mathcal{S}$, any $\mathcal{S}$-proof π can be transformed into a $\mathcal{S}$-proof π' with same end sequent and without any free-cut.*

The proof proceeds in a way similar to the classical proof of cut elimination for linear logic, but eliminating only free-cuts and verifying compatibility with our notion of nonlogical rule, in particular for the commutation cases.

First, observe that the only rules in which there is a condition on the context are the following ones: $(\forall\text{-}r)$, $(\exists\text{-}l)$, $(!\text{-}r)$, $(?\text{-}l)$, (R). These are thus the rules for which the commutation with cut steps are not straightforward. Commutations with logical rules other than $(!\text{-}r)$, $(?\text{-}l)$ are done in the standard way, as in pure linear logic:[5]

▶ **Lemma 8** (Standard commutations)**.** *Any logical rule distinct from $(!\text{-}r)$, $(?\text{-}l)$ can be commuted under a cut. If the logical rule is binary this may produce two cuts, each in a separate branch.*

For rules $(!\text{-}r)$, $(?\text{-}l)$, (R) we establish our second key lemma:

[5] Note that, for the $(\forall\text{-}r)$, $(\exists\text{-}l)$ rules, there might also be a global renaming of eigenvariables if necessary.

▶ **Lemma 9** (Key commutations). *A cut of the following form, where $?A$ is not principal for (R), can be commuted above the (R) step:*

$$\dfrac{\dfrac{\{!\Gamma, \Sigma_i \vdash \Delta_i, ?A, ?\Pi\}_{i\in I}}{!\Gamma, \Sigma' \vdash \Delta', ?A, ?\Pi}\,(R) \quad ?A, !\Gamma' \vdash ?\Pi'}{!\Gamma', \Gamma, \Sigma' \vdash \Delta', ?A, ?\Pi, ?\Pi'}\,cut$$

Similarly if (R) is replaced with $(!\text{-}r)$, with $?A$ in its RHS context, and also for the symmetric situations: cut on the LHS of the conclusion of an (R) or a $(?\text{-}l)$ step on a (non-principal) formula $!A$, with a sequent $!\Gamma' \vdash ?\Pi', !A$.

Proof. The derivation is transformed as follows:

$$\dfrac{\dfrac{!\Gamma, \Sigma_i \vdash \Delta_i, ?A, ?\Pi \quad ?A, !\Gamma' \vdash ?\Pi'}{\{!\Gamma', !\Gamma, \Sigma_i \vdash \Delta_i, ?\Pi, ?\Pi'\}_{i\in I}}\,cut}{!\Gamma', !\Gamma, \Sigma' \vdash \Delta', ?\Pi, ?\Pi'}\,(R)$$

Here if an eigenvariable in Σ_i, Δ_i happens to be free in $!\Gamma', ?\Pi'$ we rename it to avoid the collision, which is possible because by condition 2 on nonlogical rules these eigenvariables do not appear in Σ', Δ' or $!\Gamma, ?\Pi$. So the occurrence of (R) in this new subderivation is valid.

Similarly for the symmetric derivation with a cut on the LHS of the conclusion of an (R) on a formula $!A$. The analogous situations with rules $(!\text{-}r)$ and $(?\text{-}l)$ are handled in the same way, as usual in linear logic. ◀

Now we can prove the main free-cut elimination result:

Proof sketch of Thm. 7. Given a cut step c in a proof π, we call *degree* $\deg(c)$ the number of connectives and quantifiers of its cut-formula. Now the *degree* of π, $\deg(\pi)$, is the multiset of the degrees of its non-anchored cuts. We consider the usual Dershowitz-Manna ordering on multisets of natural numbers [12].[6] The proof proceeds by induction on $\deg(\pi)$. For a given degree we proceed with a sub-induction on the *height* $h(\pi)$ of the proof.

Consider a proof π of non-null degree. We want to show how to reduce it to a proof of strictly lower degree. Consider a top-most non-anchored cut c in π, i.e. such that there is no non-anchored cut above c. Let us call A the cut-formula, and (S_1) (resp. (S_2)) the rule above the left (resp. right) premise of c.

$$\dfrac{\dfrac{}{\Gamma \vdash \Delta, A}\,S_1 \quad \dfrac{}{\Sigma, A \vdash \Pi}\,S_2}{\Gamma, \Sigma \vdash \Delta, \Pi}\,c\ cut$$

Intuitively we proceed as follows: if A is not hereditarily principal in one of its premises we try to commute c with the rule along its left premise (S_1), and if not possible then commute it with the rule along its right premise (S_2), by Lemmas 6, 8 and 9. If A is hereditarily principal in both premises we proceed with a cut-elimination step, as in standard linear logic. For this second step, the delicate part is the elimination of exponential cuts, for which we use a big-step reduction. This works because the contexts in the nonlogical rules (R) are marked with ! (resp. ?) on the LHS (resp. RHS). ◀

[6] Let $M, N : \mathbb{N} \to \mathbb{N}$ be two multisets of natural numbers. Then $M < N$ if $M \neq N$ and, whenever $M(x) > N(x)$ there is some $y > x$ such that $N(y) > M(y)$. When M and N are finite, i.e. have finite support, $<$ is well-founded.

4 A variant of arithmetic in linear logic

For the remainder of this article we will consider an implementation of arithmetic in the sequent calculus based on the theory $\mathcal{A}_2^1$ of Bellantoni and Hofmann in [7]. The axioms that we present are obtained from $\mathcal{A}_2^1$ by using linear logic connectives in place of their classical analogues, calibrating the use of additives or multiplicatives in order to be compatible with the completeness and witnessing arguments that we present in Sects. 6 and 7. We also make use of free variables and the structural delimiters of the sequent calculus to control the logical complexity of nonlogical rules.

We will work in the *affine* variant of linear logic, which validates weakening: $(A \otimes B) \multimap A$. There are many reasons for this; essentially it does not have much effect on complexity while also creating a more robust proof theory. For example it induces the equivalence: $!(A \otimes B) \equiv (!A \otimes !B)$.[7]

4.1 Axiomatisation and an equivalent rule system

We consider the language $\mathcal{L}$ consisting of the constant symbol ε, unary function symbols $\mathsf{s}_0, \mathsf{s}_1$ and the predicate symbol W, together with function symbols f, g, h etc. $\mathcal{L}$-structures are typically extensions of $\mathbb{W} = \{0,1\}^*$, in which $\varepsilon, \mathsf{s}_0, \mathsf{s}_1$ are intended to have their usual interpretations. The W predicate is intended to indicate those elements of the model that are binary words (in the same way as Peano's N predicate indicates those elements that are natural numbers).

As an abbreviation, we write $W(\vec{t})$ for $\bigotimes_{i=1}^{|\vec{t}|} W(t_i)$.

▶ Remark (Interpretation of natural numbers). Notice that the set $\mathbb{N}^+$ of positive integers is $\mathcal{L}$-isomorphic to $\mathbb{W}$ under the interpretation $\{\varepsilon \mapsto 1, \mathsf{s}_0(x) \mapsto 2x, \mathsf{s}_1(x) \mapsto 2x+1\}$, so we could equally consider what follows as theories over $\mathbb{N}^+$.

The 'basic' axioms are essentially the axioms of Robinson arithmetic (or Peano Arithmetic without induction) without axioms for addition and multiplication. Let us write $\forall x^W.A$ for $\forall x.(W(x) \multimap A)$ and $\exists x^W.A$ for $\exists x.(W(x) \otimes A)$. We use the abbreviations $\forall x^{!W}$ and $\exists x^{!W}$ similarly.

▶ **Definition 10** (Basic axioms). The theory *BASIC* consists of the following axioms:

$W(\varepsilon)$	$\forall x^W.(\varepsilon \neq \mathsf{s}_0 x \otimes \varepsilon \neq \mathsf{s}_1 x)$	$\forall x^W.\mathsf{s}_0 x \neq \mathsf{s}_1 x$
$\forall x^W.W(\mathsf{s}_0 x)$	$\forall x^W, y^W.(\mathsf{s}_0 x = \mathsf{s}_0 y \multimap x = y)$	$\forall x^W.(x = \varepsilon \oplus \exists y^W.x = \mathsf{s}_0 y \oplus \exists y^W.x = \mathsf{s}_1 y)$
$\forall x^W.W(\mathsf{s}_1 x)$	$\forall x^W, y^W.(\mathsf{s}_1 x = \mathsf{s}_1 y \multimap x = y)$	$\forall x^W.(W(x) \otimes W(x))$

These axioms insist that, in any model, the set induced by $W(x)$ has the free algebra $\mathbb{W}$ as an initial segment. Importantly, there is also a form of contraction for the W predicate. We will consider theories over *BASIC* extended by induction schemata:

▶ **Definition 11** (Induction). The *(polynomial) induction* axiom schema, *PIND*, consists of the following axioms,

$$A(\varepsilon) \multimap !(\forall x^{!W}.(A(x) \multimap A(\mathsf{s}_0 x))) \multimap !(\forall x^{!W}.(A(x) \multimap A(\mathsf{s}_1 x))) \multimap \forall x^{!W}.A(x)$$

for each formula $A(x)$.

For a class Ξ of formulae, Ξ-*PIND* denotes the set of induction axioms when $A(x) \in \Xi$.

We write $I\Xi$ to denote the theory consisting of *BASIC* and Ξ-*PIND*.

[7] Notice that the right-left direction is already valid in usual linear logic, but the left-right direction requires weakening.

We use the terminology 'polynomial induction' to maintain consistency with the bounded arithmetic literature, e.g. in [9], where it is distinguished from induction on the *value* of a string (construed as a natural number). The two forms have different computational behaviour, specifically with regards to complexity, but we will restrict attention to *PIND* throughout this work, and thus may simply refer to it as 'induction'.

▶ **Proposition 12** (Equivalent rules). *BASIC is equivalent to the following set of rules,*

$$W_\varepsilon\ \frac{}{\vdash W(\varepsilon)} \qquad W_0\ \frac{}{W(t)\vdash W(\mathsf{s}_0 t)} \qquad \varepsilon_0\ \frac{}{W(t)\vdash \varepsilon \neq \mathsf{s}_0 t} \qquad \mathsf{s}_0\ \frac{}{W(s), W(t), \mathsf{s}_0 s = \mathsf{s}_0 t \vdash s = t}$$

$$inj\ \frac{}{W(t)\vdash \mathsf{s}_0 t \neq \mathsf{s}_1 t} \qquad W_1\ \frac{}{W(t)\vdash W(\mathsf{s}_1 t)} \qquad \varepsilon_1\ \frac{}{W(t)\vdash \varepsilon \neq \mathsf{s}_1 t} \qquad \mathsf{s}_1\ \frac{}{W(s), W(t), \mathsf{s}_1 s = \mathsf{s}_1 t \vdash s = t}$$

$$surj\ \frac{}{W(t)\vdash t = \varepsilon \oplus \exists y^W.t = \mathsf{s}_0 y \oplus \exists y^W.t = \mathsf{s}_1 y} \qquad W_{cntr}\ \frac{}{W(t)\vdash W(t)\otimes W(t)}$$

and PIND is equivalent to,

$$PIND\ \frac{!W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_0 a), ?\Delta \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_1 a), ?\Delta}{!W(t), !\Gamma, A(\varepsilon)\vdash A(t), ?\Delta} \tag{2}$$

where, in all cases, t varies over arbitrary terms and the eigenvariable a does not occur in the lower sequent of the PIND rule.

Note, in particular, that since this system of rules is closed under substitution of terms for free variables, free-cut elimination, Thm. 7, applies.

When converting from a *PIND* axiom instance to a rule instance (or vice-versa) the induction formula remains the same. For this reason when we consider theories that impose logical restrictions on induction we can use either interchangeably.

▶ Remark. Usually the induction axiom is also equivalent to a formulation with a designated premise for the base case:

$$\frac{!\Gamma\vdash A(\varepsilon) \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_0 a), ?\Delta \quad !W(a), !\Gamma, A(a)\vdash A(\mathsf{s}_1 a), ?\Delta}{!W(t), !\Gamma\vdash A(t), ?\Delta} \tag{3}$$

However, this is not true in the linear logic setting since the proof that (3) simulates (2) above relies on contraction on the formula $A(\varepsilon)$, which is not in general available. Therefore (3) is somewhat weaker than (2), and is in fact equivalent to a version of the induction axiom with $!A(\varepsilon)$ in place of $A(\varepsilon)$. This distinction turns out to be crucial in Sect. 6, namely when proving the convergence of functions defined by predicative recursion on notation.

4.2 Provably convergent functions

As in the work of Bellantoni and Hofmann [7] and Leivant before [20], our model of computation is that of Herbrand-Gödel style *equational specifications*. These are expressive enough to define every partial recursive function, which is the reason why we also need the W predicate to have a meaningful notion of 'provably convergent function'.

▶ **Definition 13** (Equational specifications and convergence). An *equational specification* (ES) is a set of equations between terms. We say that an ES is *coherent* if the equality between any two distinct ground terms cannot be proved by equational logic.

The *convergence statement* $Conv(f, \mathcal{E})$ for an equational specification $\mathcal{E}$ and a function symbol f (that occurs in $\mathcal{E}$) is the following formula:

$$\bigotimes_{A\in\mathcal{E}} !\forall\vec{x}.A \ \multimap\ \forall\vec{x}^{!W}.W(f(\vec{x}))$$

The notion of coherence appeared in [20] and it is important to prevent a convergence statement from being a vacuous implication. In this work we will typically consider only coherent ESs, relying on the following result which is also essentially in [20]:

▶ **Proposition 14.** *The universal closure of a coherent ES $\mathcal{E}$ has a model satisfying BASIC + PIND.*

One issue is that a convergence statement contains universal quantifiers, which is problematic for the extraction of functions by the witness function method later on. We avoid this problem by appealing to the deduction theorem and further invertibility arguments:

Let us write $\overline{\mathcal{E}}$ for the closure of a specification $\mathcal{E}$ under substitution of terms for free variables.

▶ **Lemma 15.** *A system $\mathcal{S}$ proves $Conv(f, \mathcal{E})$ if and only if $\mathcal{S} \cup \overline{\mathcal{E}}$ proves $!W(\vec{a}) \vdash W(f(\vec{a}))$.*

Proof sketch. By deduction, Thm. 3, and invertibility arguments. ◀

Notice that the initial rules from $\overline{\mathcal{E}}$ are also closed under term substitution, and so compatible with free-cut elimination, and that $\overline{\mathcal{E}}$ and $W(\vec{a}) \vdash W(f(\vec{a}))$ are free of negation and universal quantifiers.

4.3 W-guarded quantifiers, rules and cut-reduction cases

We consider a quantifier hierarchy here analogous to the arithmetical hierarchy, where each class is closed under positive multiplicative operations. In the scope of this work we are only concerned with the first level:

▶ **Definition 16.** We define $\Sigma_0^{W^+}$ as the class of multiplicative formulae that are free of quantifiers where W occurs positively.[8] The class $\Sigma_1^{W^+}$ is the closure of $\Sigma_0^{W^+}$ by $\exists$, $\parr$ and $\otimes$.

For the remainder of this article we mainly work with the theory $I\Sigma_1^{W^+}$, i.e. *BASIC* + $\Sigma_1^{W^+}$*-PIND*.

It will be useful for us to work with proofs using the 'guarded' quantifiers $\forall x^W$ and $\exists x^W$ in place of their unguarded counterparts, in particular to carry out the argument in Sect. 7. Therefore we define the following rules, which are already derivable:

$$\frac{\Gamma, W(a) \vdash \Delta, A(a)}{\Gamma \vdash \Delta, \forall x^W.A(x)} \qquad \frac{\Gamma, A(t) \vdash \Delta}{\Gamma, W(t), \forall x^W A(x) \vdash \Delta} \qquad \frac{\Gamma, W(a), A(a) \vdash \Delta}{\Gamma, \exists x^W A(x) \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A(t)}{\Gamma, W(t) \vdash \Delta, \exists x^W.A(x)}$$

We now show that these rules are compatible with free-cut elimination.

▶ **Proposition 17.** *Any cut between the principal formula of a quantifier rule above and the principal formula of a logical step is reducible.*

Proof. For a cut on $\forall x^W.A(x)$, the reduction is obtained by performing successively the two reduction steps for the $\forall$ and $\multimap$ connectives. The case of $\exists x^W A(x)$ is similar. ◀

▶ **Corollary 18** (Free-cut elimination for guarded quantifiers)**.** *Given a system $\mathcal{S}$, any $\mathcal{S}$-proof π using $\exists x^W$ and $\forall x^W$ rules can be transformed into free-cut free form.*

As a consequence of this Corollary observe that any $I\Sigma_1^{W^+}$-proof can be transformed into a proof which is free-cut free and whose formulas contain only $\exists x^W$ quantifiers.

[8] Since our proof system is in De Morgan normal form, this is equivalent to saying that there is no occurrence of $W^\perp$.

5 Bellantoni-Cook characterisation of polynomial-time functions

We recall the Bellantoni-Cook algebra BC of functions defined by *safe* (or *predicative*) recursion on notation [6]. These will be employed for proving both the completeness (all polynomial time functions are provably convergent) and the soundness result (all provably total functions are polynomial time) of $I\Sigma_1^{W^+}$. We consider function symbols f over the domain $\mathbb{W}$ with sorted arguments $(\vec{u};\vec{x})$, where the inputs $\vec{u}$ are called *normal* and $\vec{x}$ are called *safe*.

▶ **Definition 19** (BC programs)**.** BC is the set of functions generated as follows:

1. The constant functions ε^k which takes k arguments and outputs $\varepsilon \in \mathbb{W}$.
2. The projection functions $\pi_k^{m,n}(x_1,\ldots,x_m;x_{m+1},\ldots,x_{m+n}) := x_k$ for $n,m \in \mathbb{W}$ and $1 \leq k \leq m+n$.
3. The successor functions $\mathsf{s}_i(;x) := xi$ for $i = 0,1$.
4. The predecessor function $p(;x) := \begin{cases} \varepsilon & \text{if } x = \varepsilon \\ x' & \text{if } x = x'i \end{cases}$.
5. The conditional function

$$C(;\varepsilon,y_\varepsilon,y_0,y_1) := y_\varepsilon \quad C(;x0,y_\varepsilon,y_0,y_1) := y_0 \quad C(;x1,y_\varepsilon,y_0,y_1) := y_1$$

6. Predicative recursion on notation (PRN). If g,h_0,h_1 are in BC then so is f defined by,

$$\begin{array}{rcl} f(0,\vec{v};\vec{x}) & := & g(\vec{v};\vec{x}) \\ f(\mathsf{s}_i u,\vec{v};\vec{x}) & := & h_i(u,\vec{v};\vec{x},f(u,\vec{v};\vec{x})) \end{array}$$

   for $i = 0,1$, so long as the expressions are well-formed.
7. Safe composition. If $g,\vec{h},\vec{h'}$ are in BC then so is f defined by,

$$f(\vec{u};\vec{x}) \quad := \quad g(\vec{h}(\vec{u};);\vec{h'}(\vec{u};\vec{x}))$$

   so long as the expression is well-formed.

We will implicitly identify a BC function with the equational specification it induces. The main property of BC programs is:

▶ **Theorem 20** ([6])**.** *The class of functions representable by BC programs is* **FP**.

Actually this property remains true if one replaces the PRN scheme by the following more general simultaneous PRN scheme [8]:

$(f^j)_{1\leq j\leq n}$ are defined by simultaneous PRN scheme from $(g^j)_{1\leq j\leq n}$, $(h_0^j,h_1^j)_{1\leq j\leq n}$ if for $1 \leq j \leq n$ we have:

$$\begin{array}{rcl} f^j(0,\vec{v};\vec{x}) & := & g^j(\vec{v};\vec{x}) \\ f^j(\mathsf{s}_i u,\vec{v};\vec{x}) & := & h_i^j(u,\vec{v};\vec{x},\vec{f}(u,\vec{v};\vec{x})) \end{array}$$

for $i = 0,1$, so long as the expressions are well-formed.

Consider a well-formed expression t built from function symbols and variables. We say that a variable y occurs *hereditarily safe* in t if, for every subexpression $f(\vec{r};\vec{s})$ of t, the terms in $\vec{r}$ do not contain y. For instance y occurs hereditarily safe in $f(u;y,g(v;y))$, but not in $f(g(v;y);x)$.

▶ **Proposition 21** (Properties of BC programs)**.** *We have the following properties:*

1. *The identity function is in BC.*

2. *Let t be a well-formed expression built from BC functions and variables, denote its free variables as $\{u_1, \ldots, u_n, x_1, \ldots, x_k\}$, and assume for each $1 \leq i \leq k$, x_i is hereditarily safe in t. Then the function $f(u_1, \ldots, u_n; x_1, \ldots, x_k) := t$ is in BC.*
3. *If f is a BC function, then the function $g(\vec{u}, v; \vec{x})$ defined as $f(\vec{u}; v, \vec{x})$ is also a BC program.*

6 Convergence of Bellantoni-Cook programs in $I\Sigma_1^{W^+}$

In this section we show that $I\Sigma_0^{W^+}$, and so also $I\Sigma_1^{W^+}$, proves the convergence of any equational specification induced by a BC program, hence any function in **FP**. The underlying construction of the proof here is similar in spirit to those occurring in [11] and [20]. In fact, like in those works, only quantifier-free positive induction is required, but here we moreover must take care to respect additive and multiplicative behaviour of linear connectives.

We will assume the formulation of BC programs with regular PRN, not simultaneous PRN.

▶ **Theorem 22.** *If $\mathcal{E}$ is a BC program defining a function f, then $I\Sigma_0^{W^+}$ proves $Conv(f, \mathcal{E})$.*

Proof sketch. We appeal to Lemma 15 and show that $\overline{\mathcal{E}} \cup I\Sigma_0^{W^+}$ proves $\forall \vec{u}^{!W}.\forall \vec{x}^{W}.W(f(\vec{u};\vec{x}))$. We proceed by induction on the structure of a BC program for f, and sketch only the key cases here.

Suppose $f(u, \vec{v}; \vec{x})$ is defined by PRN from functions $g(\vec{v}; \vec{x}), h_i(u, \vec{v}; \vec{x}, y)$. From the inductive hypothesis for g, we construct the following proof,

$$\beta\,\dfrac{\alpha\,\dfrac{\vdash \forall \vec{v}^{!W}.\forall \vec{x}^{W}.W(g(\vec{v};\vec{x}))}{!W(\vec{v}), W(\vec{x}) \vdash W(g(\vec{v};\vec{x}))}}{!W(\vec{v}), W(\vec{x}) \vdash W(f(\varepsilon, \vec{v}; \vec{x}))} \tag{4}$$

where α is purely logical and β is obtained from $\overline{\mathcal{E}}$ and equality. We also construct the proofs,

$$\otimes\text{-}l\,\dfrac{cntr\text{-}l\,\dfrac{\otimes\text{-}r\,\dfrac{id\,\dfrac{}{W(\vec{x}) \vdash W(\vec{x})} \quad \beta\,\dfrac{\alpha\,\dfrac{\vdash \forall u^{!W}, \vec{v}^{!W}.\forall \vec{x}^{W}, y^{W}.W(h_i(u,\vec{v};\vec{x},y))}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(u,\vec{v};\vec{x})) \vdash W(h_i(u,\vec{v};\vec{x},f(u,\vec{v};\vec{x})))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(u,\vec{v};\vec{x})) \vdash W(f(\mathsf{s}_i u,\vec{v};\vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}), W(\vec{x}), W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a,\vec{v};\vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}), W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a,\vec{v};\vec{x}))}}{!W(a), !W(\vec{v}), W(\vec{x}) \otimes W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i a,\vec{v};\vec{x}))} \tag{5}$$

from the inductive hypotheses for h_i, where α and β are similar to before. Finally we compose these proofs as follows:

$$\forall\text{-}r\,\dfrac{cntr\text{-}l\,\dfrac{cut\,\dfrac{!W(\vec{v}), W(\vec{x}) \vdash W(f(\varepsilon,\vec{v};\vec{x})) \qquad wk\,\dfrac{\otimes\text{-}inv\,\dfrac{PIND\,\dfrac{\{!W(a), !W(\vec{v}), W(\vec{x})W(f(a,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(\mathsf{s}_i u,\vec{v};\vec{x}))\}_i}{!W(u), !W(\vec{v}), W(\vec{x}) \otimes W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(\vec{x}) \otimes W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}), W(f(\varepsilon,\vec{v};\vec{x})) \vdash W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), !W(\vec{v}), W(\vec{x}), W(\vec{x}) \vdash W(f(u,\vec{v};\vec{x}))}}{!W(u), !W(\vec{v}), W(\vec{x}) \vdash W(f(u,\vec{v};\vec{x}))}}{\vdash \forall u^{!W}, \vec{v}^{!W}.\forall \vec{x}^{W}.W(f(u,\vec{v};\vec{x}))}$$

for $i = 0, 1$, where the steps $\otimes$-inv are obtained from invertibility of $\otimes$-l.

Safe compositions are essentially handled by many cut steps, using α and β like derivations again and, crucially, left-contractions on both $!W$ and W formulae.[9] The initial functions are routine. ◀

7 Witness function method

We now prove the converse to the last section: any provably convergent function in $I\Sigma_1^{W^+}$ is polynomial-time computable,

using the witness function method (WFM) [9].

The WFM differs from realisability and Dialectica style witnessing arguments mainly since it does not require functionals at higher type. Instead a translation is conducted directly from a proof in De Morgan normal form, i.e. with negation pushed to the atoms, relying on classical logic.

The combination of De Morgan normalisation and free-cut elimination plays a similar role to the double-negation translation, and this is even more evident in our setting where the transformation of a classical proof to free-cut free form can be seen to be a partial 'constructivisation' of the proof. As well as eliminating the (nonconstructive) occurrences of the $\forall$-right rule, as usual for the WFM, the linear logic refinement of the logical connectives means that right-contraction steps are also eliminated. This is important due to the fact that we are in a setting where programs are equational specifications, not formulae (as in bounded arithmetic [9]) or combinatory terms (as in applicative theories [11]), so we cannot in general decide atomic formulae.

7.1 The translation

We will associate to each (free-cut free) proof of a convergence statement in $I\Sigma_1^{W^+}$ a function on $\mathbb{W}$ defined by a BC program. In the next section we will show that this function satisfies the equational specification of the convergence statement.

▶ **Definition 23** (Typing). To each $(\forall, ?)$-free W^+-formula A we associate a sorted tuple of variables $\mathtt{t}(A)$, intended to range over $\mathbb{W}$, as follows:

$$\begin{array}{llll} \mathtt{t}(W(t)) := (;x^{W(t)}) & \mathtt{t}(s \neq t) := (;x^{s\neq t}) & \mathtt{t}(A \star B) := (\vec{u},\vec{v};\vec{x},\vec{y}) \\ \mathtt{t}(s = t) := (;x^{s=t}) & \mathtt{t}(!A) := (\vec{u},\vec{x};) & \mathtt{t}(\exists x^W.A) := (\vec{u};\vec{x},y) \end{array}$$

where $\mathtt{t}(A) = (\vec{u};\vec{x})$, $\mathtt{t}(B) = (\vec{v};\vec{y})$ and $\star \in \{\parr, \otimes, \oplus, \&\}$.

For a sorted tuple $(u_1, \ldots u_m; x_1, \ldots, x_n)$ we write $|(\vec{u};\vec{x})|$ to denote its length, i.e. $m+n$. This sorted tuple corresponds to input variables, normal and safe respectively.

Let us fix a particular (coherent) equational specification $\mathcal{E}(f)$. Rather than directly considering $I\Sigma_1^{W^+}$-proofs of $\mathit{Conv}(f,\mathcal{E})$, we will consider a $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ sequent proof of $!W(\vec{x}) \vdash W(f(\vec{x}))$, under Lemma 15. Free-cut elimination crucially yields strong regularity properties for proofs:

▶ **Proposition 24** (Freedom). *A free-cut free $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ sequent proof of $!W(\vec{x}) \vdash W(f(\vec{x}))$ is:*

1. *Free of any negative occurrences of W.*
2. *Free of any $\forall$ symbols.*

[9] In the latter case, strictly speaking, we mean cuts against W_{cntr}.

3. *Free of any ? symbols.*
4. *Free of any $\oplus$ or & steps.*[10]

For this reason we can assume that $\mathtt{t}$ is well-defined for all formulae occurring in a free-cut free proof of convergence, and so we can proceed with the translation from proofs to BC programs.

▶ **Definition 25** (Translation). We give a translation from a free-cut free $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$ proof π, satisfying properties 1, 2, 3, 4 of Prop. 24 above, of a sequent $\Gamma \vdash \Delta$ to BC programs for a tuple of functions $\vec{f}^{\pi}$ with arguments $\mathtt{t}\,(\bigotimes \Gamma)$ such that $|\vec{f}^{\pi}| = |\mathtt{t}\,(\parr \Delta)\,|$.

The translation is by induction on the structure of π, so we proceed by inspection of its final step.

If π is an instance of the initial rules $W_\varepsilon, \varepsilon^0, \varepsilon^1, \mathsf{s}_0, \mathsf{s}_1$ or *inj* then $\vec{f}^{\pi}$ is simply the constant function ε (possibly with dummy inputs as required by $\mathtt{t}$). If π is an $\overline{\mathcal{E}}$ or $=$ initial step it is also translated simply to ε. The initial steps W_0, W_1, *surj* and W_{cntr} are translated to $\mathsf{s}_0(;x), \mathsf{s}_1(;x), (\varepsilon, p(;x), p(;x))$ and $(id(;x), id(;x))$ respectively. Finally, suppose π is a logical initial step. If π is an instance of *id*, i.e. $p \vdash p$, then we translate it to *id*. Notice that, if π is an instance of $\bot$-l (i.e. $p, p^\bot \vdash$) or $\bot$-r (i.e. $\vdash p, p^\bot$) then p must be an equality $s = t$ for some terms s, t, since p must be atomic and, by assumption, W does not occur negatively. Therefore π is translated to tuples of ε as appropriate.

Now we consider the inductive cases. If π ends with a $\otimes$-r or $\parr$-l step then we just rearrange the tuple of functions obtained from the inductive hypothesis. If π consists of a subproof π' ending with a $\otimes$-l or $\parr$-r-step, then $\vec{f}^{\pi}$ is exactly $\vec{f}^{\pi'}$. By assumption, there are no $\oplus$, &, ? or $\forall$ steps occurring in π, and if π consists of a subproof π' followed by a $\exists$-l step then $\vec{f}^{\pi}$ is exactly the same as $\vec{f}^{\pi'}$, under possible reordering of the tuple.

Suppose π consists of a subproof π' followed by a $\exists$-r step,

$$\exists\text{-}r\ \frac{\Gamma \vdash \Delta, A(t)}{\Gamma, W(t) \vdash \Delta, \exists x^W.A(x)}$$

so by the inductive hypothesis we have functions $\vec{f}^{\Delta}, \vec{f}^{A(t)}$ with arguments $(\vec{u};\vec{x}) = \mathtt{t}(\bigotimes \Gamma)$. We define $\vec{f}^{\pi}(\vec{u};\vec{x},y)$ as $\Big(\vec{f}^{\Delta}(\vec{u};\vec{x}), id(;y), \vec{f}^{A(t)}(\vec{u};\vec{x})\Big)$.

If π consists of a subproof π' followed by a !-r step then $\vec{f}^{\pi}$ is exactly the same as $\vec{f}^{\pi'}$. If π ends with a !-l step then we just appeal to Prop. 21 to turn a safe input into a normal input.

Since there are no ? symbols in π, we can assume also that there are no *cntr*-r steps in π.[11]

If π ends with a *cntr*-l step then we duplicate some normal inputs of the functions obtained by the inductive hypothesis.

If π ends with a *cut* step whose cut-formula is free of modalities, then it can be directly translated to a safe composition of functions obtained by the inductive hypothesis, by relying on Prop. 21. Otherwise, the cut-formula must be of the form $!W(t)$ since it must directly descend from the left-hand side of an induction step, by free-cut freeness. Since the cut is

[10] Because of the *surj* rule, the proof may still contain $\oplus$ symbols, but these must be direct ancestors of some cut-step by free-cut freeness.

[11] Again, this is crucially important, since we cannot test the equality between arbitrary terms in the presence of nonlogical function symbols.

anchored, we can also assume that the cut formula is principal on the other side, i.e. π ends as follows:

$$\dfrac{\dfrac{!\Gamma \vdash W(t)}{!\Gamma \vdash !W(t)}\,{\scriptstyle !\text{-}r} \quad !W(t), \Sigma \vdash \Delta}{!\Gamma, \Sigma \vdash \Delta}\,{\scriptstyle cut}$$

where we assume there are no side-formulae on the right of the end-sequent of the left subproof for the same reason as *cntr-r*: π does not contain any occurrences of ?. By the inductive hypothesis we have functions $g(\vec{u};)$ and $\vec{h}(v, \vec{w}; \vec{x})$ where $\vec{u}$, v and $(\vec{w}; \vec{x})$ correspond to $!\Gamma$, $!W(t)$ and Σ respectively. We construct the functions $\vec{f^{\pi}}$ as follows:

$$\vec{f^{\pi}}(\vec{u}, \vec{w}; \vec{x}) \quad := \quad \vec{h}(g(\vec{u};), \vec{w}; \vec{x})$$

Notice, again, that all safe inputs on the left occur hereditarily safe on the right, and so these expressions are definable in BC by Prop. 21.

If π ends with a *wk-r* step then we just add a tuple of constant functions $\vec{\varepsilon}$ of appropriate length as dummy functions. If π ends with a *wk-l* step then we just add dummy inputs of appropriate length.

Finally, suppose π ends with a *PIND* step. Since there are no occurrences of ? in π we can again assume that there are no side formulae on the right of any induction step. Thus π ends as follows:

$$PIND\,\dfrac{!W(a), !\Gamma, A(a) \vdash A(\mathsf{s}_0 a) \quad !W(a), !\Gamma, A(a) \vdash A(\mathsf{s}_1 a)}{!W(t), !\Gamma, A(\varepsilon) \vdash A(t)}$$

By the inductive hypothesis we have functions $\vec{g}^0(u, \vec{v}; \vec{x})$ and $\vec{g}^1(u, \vec{v}; \vec{x})$ with u, $\vec{v}$ and $\vec{x}$ corresponding to $!W(a)$, $!\Gamma$ and $A(a)$ respectively. We define $\vec{f^{\pi}}$ by simultaneous predicative recursion on notation as follows:

$$\begin{array}{rcl} \vec{f^{\pi}}(\varepsilon, \vec{v}; \vec{x}) & := & \vec{x} \\ \vec{f^{\pi}}(\mathsf{s}_i u, \vec{v}; \vec{x}) & := & \vec{g^i}(u, \vec{v}; \vec{f^{\pi}}(u, \vec{v}; \vec{x})) \end{array}$$

The induction step above is the reason why we enrich the usual BC framework with a simultaneous version of PRN.

7.2 Witness predicate and extensional equivalence of functions

Now that we have seen how to extract BC functions from proofs, we show that these functions satisfy the appropriate semantic properties, namely the equational program $\mathcal{E}$ we started with. For this we turn to a quantifier-free *classical* theory, in a similar fashion to PV for S^1_2 in [9] or system T in Gödel's Dialectica interpretation. This is adequate since we only care about extensional properties of extracted functions at this stage.

We could equally use a realisability approach, as done in e.g. [11] and other works in applicative theories: since the formulae we deal with are essentially positive there is not much difference between the two approaches. Indeed here the witness predicate plays the same role as the realisability predicate in other works.

Let *IQF* be the classical theory over the language $\{\varepsilon, \mathsf{s}_0, \mathsf{s}_1, (f^k_i)\}$ obtained from the axioms $\varepsilon, \mathsf{s}_0, \mathsf{s}_1$, *inj*, *surj* and *PIND* by dropping all relativisations to W (or $!W$), replacing all linear connectives by their classical counterparts, and restricting induction to only quantifier-free formulae.

▶ **Definition 26** (Witness predicate). For formulae A, B of $I\Sigma_1^{W^+}$ satisfying properties 1, 2, 3 of Prop. 24, we define the following 'witness' predicate as a quantifier-free formula of IQF:

$$\begin{array}{rcl} Wit_{W(t)}(a) & := & a = t \\ Wit_{s=t}(a) & := & s = t \\ Wit_{s\neq t}(a) & := & s \neq t \\ Wit_{!A}(\vec{a}^A) & := & Wit_A(\vec{a}^A) \end{array} \qquad \begin{array}{rcl} Wit_{A\bullet B}(\vec{a}^A, \vec{a}^B) & := & Wit_A(\vec{a}^A) \vee Wit_B(\vec{a}^B) \\ Wit_{A\circ B}(\vec{a}^A, \vec{a}^B) & := & Wit_A(\vec{a}^A) \wedge Wit_B(\vec{a}^B) \\ Wit_{\exists x^W A}(a, \vec{a}^A) & := & Wit_A(\vec{a}^A, a) \end{array}$$

where $\bullet \in \{\parr, \oplus\}$, $\circ \in \{\otimes, \&\}$, $|\vec{a}^A| = |\mathtt{t}(A)|$ and $|\vec{a}^B| = |\mathtt{t}(B)|$.

Notice that, unlike in the bounded arithmetic setting where the W predicate is redundant (since variables are tacitly assumed to range over W), we do not parametrise the witness predicate by an assignment to the free variables of a formula. Instead these dependencies are taken care of by the explicit occurrences of the W predicate in $I\Sigma_1^{W^+}$.

▶ **Lemma 27.** *Let π be a free-cut free proof in $\overline{\mathcal{E}} \cup I\Sigma_1^{W^+}$, satisfying properties 1, 2, 3, 4 of Prop. 24, of a sequent $\Gamma \vdash \Delta$. Let $\mathcal{E}^\pi$ denote the BC program for $\vec{f}^\pi$.*[12] *Then IQF proves:*

$$\left(\forall\mathcal{E} \wedge \forall\mathcal{E}^\pi \wedge Wit_{\bigotimes\Gamma}(\vec{a})\right) \to Wit_{\parr\Delta}(\vec{f}^\pi(\vec{a}))$$

where $\forall\mathcal{E}$ and $\forall\mathcal{E}^\pi$ denote the universal closures of $\mathcal{E}$ and $\mathcal{E}^\pi$ respectively.

Proof sketch. By structural induction on π, again, following the definition of $\vec{f}^\pi$.[13] ◀

Finally, we arrive at our main result, providing a converse to Thm. 22.

▶ **Theorem 28.** *For any coherent equational specification $\mathcal{E}$, if $I\Sigma_1^{W^+}$ proves $Conv(f, \mathcal{E})$ then there is a polynomial-time function g on $\mathbb{W}$ (of same arity as f) satisfying $\mathcal{E}[g/f]$.*

Proof sketch. Follows from Lemmas 15 and 27, Dfn. 26 and coherence of $\mathcal{E}$, cf. 14. ◀

8 Conclusions

As mentioned in the introduction, our motivation for this work is to commence a proof-theoretic study of first-order theories in linear logic, in particular from the point of view of complexity. To this end we proved a general form of 'free-cut elimination' that generalises forms occurring in prior works, e.g. [22]. We adapted an arithmetic of Bellantoni and Hofmann in [7] to the linear logic setting, and used the free-cut elimination result, via the witness function method [9], to prove that a fragment of this arithmetic characterises **FP**.

From the point of view of constructing theories for complexity classes, the choice of linear logic and witness function method satisfies two particular desiderata:

1. Complexity is controlled by 'implicit' means, not explicit bounds.
2. Extraction of programs relies on functions of only ground type.

From this point of view, a relevant related work is that of Cantini [11], based on an *applicative theory*, which we recently became aware of. The main difference here is the choice of model of computation: Cantini uses terms of combinatory logic, whereas we use equational specifications (ESs). As we have mentioned, this choice necessitates a different

[12] We assume that the function symbols occurring in $\mathcal{E}^\pi$ are distinct from those occurring in $\mathcal{E}$.

[13] Notice that, since we are in a classical theory, the proof of the above lemma can be carried out in an arbitrary model, by the completeness theorem, greatly easing the exposition.

proof-theoretic treatment, in particular since equality between terms is not decidable in the ES framework, hindering any constructive interpretation of the right-contraction rule. This is why Bellantoni and Hofmann require a double-negation translation into intuitionistic logic and the use of functionals at higher types, and why Leivant disregards classical logic altogether in [20]. Notice that this is precisely why our use of linear logic is important: the control of ? occurrences in a proof allows us to sidestep this problem. At the same time we are able to remain in a classical linear setting. We do not think that either model of computation is better, but rather that it is interesting to observe how such a choice affects the proof-theoretic considerations at hand.

Most works on the relationships between linear logic and complexity fit in the approach of the proofs-as-programs correspondence and study variants of linear logic with weak exponential modalities [17] [15] [18]. However, Terui considers a naïve set theory [27] that also characterises **FP** and is based on *light linear logic*.[14] His approach relies on functionals at higher type, using the propositional fragment of the logic to type the extracted functionals. Another work using linear logic to characterize complexity classes by using convergence proofs is [19] but it is tailored for second-order logic. The status of first-order theories is more developed for other substructural logics, for instance *relevant logic* [13], although we do not know of any works connecting such logics to computational complexity.

Concerning the relationship between linear logic and safe recursion, we note that an embedding of a variant of safe recursion into light linear logic has been carried studied in [25], but this is in the setting of functional computation and is quite different from the present approach. Observe, however, that a difficulty in this setting was the nonlinear treatment of safe arguments which here we manage by including in our theory an explicit contraction axiom for W.

We have already mentioned the work of Bellantoni and Hofmann [7], which was somewhat the inspiration behind this work. Our logical setting is very similar to theirs, under a certain identification of symbols, but there is a curious disconnect in our use of the additive disjunction for the *surj* axiom. They rely on just one variant of disjunction. As we said, they rely on a double-negation translation and thus functionals at higher-type.

In further work we would like to apply the free-cut elimination theorem to theories based on other models of computation, namely the formula model employed in bounded arithmetic [9]. We believe that the witness function method could be used at a much finer level in this setting,[15] and extensions of the theory for other complexity classes, e.g. the polynomial hierarchy, might be easier to obtain.

The problem of right-contraction seems to also present in work by Leivant [20], which uses equational specifications, where the restriction to positive comprehension to characterise polynomial-time only goes through in an intuitionistic setting. It would be interesting to see if a linear logic refinement could reproduce that result in the classical setting, as we did here.

[14] He also presents a cut-elimination result but, interestingly, it is entirely complementary to that which we present here: he obtains full cut-elimination since he works in a system without full exponentials and with Comprehension as the only nonlogical rule. Since the latter admits a cut-reduction step, the former ensures that cut-elimination eventually terminates by a *height-based* argument, contrary to our argument that analyses the *degrees* of cut-formulae.

[15] One reason for this is that atomic formulae are decidable, and so we can have more freedom with the modalities in induction steps.

References

1 Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. Log. Comput.*, 2(3):297–347, 1992. doi:10.1093/logcom/2.3.297.

2 Arnon Avron. The semantics and proof theory of linear logic. *Theor. Comput. Sci.*, 57:161–184, 1988.

3 David Baelde. Least and greatest fixed points in linear logic. *ACM Trans. Comput. Log.*, 13(1):2, 2012.

4 Patrick Baillot and Anupam Das. Free-cut elimination in linear logic and an application to a feasible arithmetic. Preprint, 2016. URL: https://hal.archives-ouvertes.fr/hal-01316754.

5 Arnold Beckmann and Samuel R. Buss. Corrected upper bounds for free-cut elimination. *Theor. Comput. Sci.*, 412(39):5433–5445, 2011.

6 Stephen Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, 2:97–110, 1992.

7 Stephen Bellantoni and Martin Hofmann. A new "feasible" arithmetic. *J. Symb. Log.*, 67(1):104–116, 2002.

8 Stephen J. Bellantoni. *Predicative Recursion and Computational Complexity*. PhD thesis, University of Toronto, 1992.

9 Samuel R Buss. *Bounded arithmetic*, volume 86. Bibliopolis, 1986.

10 Samuel R Buss. An introduction to proof theory. *Handbook of proof theory*, 137:1–78, 1998.

11 Andrea Cantini. Polytime, combinatory logic and positive safe induction. *Arch. Math. Log.*, 41(2):169–189, 2002.

12 Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, August 1979. doi:10.1145/359138.359142.

13 Harvey Friedman and Robert K. Meyer. Whither relevant arithmetic? *J. Symb. Log.*, 57(3):824–831, 1992. doi:10.2307/2275433.

14 Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. doi:10.1016/0304-3975(87)90045-4.

15 Jean-Yves Girard. Light linear logic. In *Logical and Computational Complexity. Selected Papers. LCC'94.*, pages 145–176, 1994. doi:10.1007/3-540-60178-3_83.

16 Jean-Yves Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.

17 Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.

18 Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.

19 Marc Lasson. Controlling program extraction in light logics. In *Typed Lambda Calculi and Applications – 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings*, volume 6690 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2011.

20 Daniel Leivant. A foundational delineation of poly-time. *Inf. Comput.*, 110(2):391–420, 1994.

21 Daniel Leivant. Intrinsic theories and computational complexity. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC'94, Indianapolis, Indiana, USA, 13-16 October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 1995.

22 Patrick Lincoln, John C. Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Ann. Pure Appl. Logic*, 56(1-3):239–311, 1992.

23 Jean-Yves Marion. Actual arithmetic and feasibility. In *Proceedings of Computer Science Logic (CSL 2001)*, volume 2142 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2001.

24 Dale Miller. Overview of linear logic programming. In Thomas Ehrhard, editor, *Linear Logic in Computer Science*, pages 316–119. Cambridge University Press, 2004.

25 Andrzej S. Murawski and C.-H. Luke Ong. On an interpretation of safe recursion in light affine logic. *Theor. Comput. Sci.*, 318(1-2):197–223, 2004.

26 G. Takeuti. *Proof Theory*. North-Holland, Amsterdam, 1987. and ed.

27 Kazushige Terui. Light affine set theory: A naive set theory of polynomial time. *Studia Logica*, 77(1):9–40, 2004.

The Relational Model Is Injective for Multiplicative Exponential Linear Logic

Daniel de Carvalho

Innopolis University, Russia
d.carvalho@innopolis.ru

Abstract

We prove a completeness result for Multiplicative Exponential Linear Logic (MELL): we show that the relational model is injective for MELL proof-nets, i.e. the equality between MELL proof-nets in the relational model is exactly axiomatized by cut-elimination.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Linear Logic, Denotational semantics, Proof-nets

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.41

1 Introduction

In the seminal paper by Harvey Friedman [11], it has been shown that equality between simply-typed lambda terms in the full typed structure $\mathcal{M}_X$ over an infinite set X is completely axiomatized by β and η: we have $\mathcal{M}_X \vDash v = u \Leftrightarrow v \simeq_{\beta\eta} u$. A natural problem is to know whether a similar result could be obtained for Linear Logic.

Such a result can be seen as a "separation" theorem. To obtain such separation theorems, it is a prerequesite to have a "canonical" syntax. When Jean-Yves Girard introduced Linear Logic (LL) [12], he not only introduced a sequent calculus system but also "proof-nets". Indeed, as for LJ and LK (sequent calculus systems for intuitionnistic and classical logic, respectively), different proofs in LL sequent calculus can represent "morally" the same proof: proof-nets were introduced to find a unique representative for these proofs.

The technology of proof-nets was completely satisfactory for the multiplicative fragment without units.[1] For proof-nets having additives, contractions or weakenings, it was easy to exhibit different proof-nets that should be identified. Despite some flaws, the discovery of proof-nets was striking. In particular, Vincent Danos proved by syntactical means in [3] the confluence of these proof-nets for the Multiplicative Exponential Linear Logic fragment (MELL). For additives, the problem to have a satisfactory notion of proof-net has been addressed in [15]. For MELL, a "new syntax" was introduced in [4]. In the original syntax, the following properties of the weakening and of the contraction did not hold:

- the associativity of the contraction;
- the neutrality of the weakening for the contraction;
- the contraction and the weakening as morphisms of coalgebras.

But they hold in the new syntax; at least for MELL, we got a syntax that was a good candidate to deserve to be considered as being "canonical". Then trying to prove that any two

[1] For the multiplicative fragment with units, it has been recently shown [14] that, in some sense, no satisfactory notion of proof-net can exist. Our proof-nets have no jump, so they identify too many sequent calculus proofs, but not more than the relational semantics.

© Daniel de Carvalho;
licensed under Creative Commons License CC-BY

(η-expanded) MELL proof-nets that are equal in some denotational semantics are β-joinable has become sensible and had at least the two following motivations:

- to prove the canonicity of the "new syntax" (if we quotient more normal proof-nets, then we would identify proof-nets having different semantics);
- to prove by semantics means the confluence (if a proof-net reduces to two cut-free proof-nets, then they have the same semantics, so they would be β-joinable, hence equal).

The problem of *injectivity*[2] of the denotational semantics for MELL, which is the question whether equality in the denotational semantics between (η-expanded) MELL proof-nets is exactly axiomatized by cut-elimination or not, can be seen as a study of the separation property with a semantic approach. The first work on the study of this property in the framework of proof-nets is [16] where the authors deal with the translation into LL of the pure λ-calculus; it has been studied more recently for the intuitionistic multiplicative fragment of LL [17] and for differential nets [18]. For Parigot's $\lambda\mu$-calculus, see [5] and [22].

Finally the precise problem of injectivity for MELL has been adressed by Lorenzo Tortora de Falco in his PhD thesis [23] and in [24] for the (multiset based) coherence semantics and the multiset based relational semantics. He gave partial results and counter-examples for the coherence semantics: the (multiset based) coherence semantics is not injective for MELL. Also, it was conjectured that the relational model is injective for MELL. We prove the conjecture in the present paper.

In [24], a proof of the injectivity of the relational model is given for a weak fragment. But despite many efforts ([23], [24], [1], [19], [18], [20]...), all the attempts to prove the conjecture failed up to now. New progress was made in [9], where it has been proved that the relational semantics is injective for "connected" MELL proof-nets. Still, there, "connected" is understood as a very strong assumption, the set of "connected" MELL proof-nets contains the fragment of MELL defined by removing weakenings and units. Actually [9] proved a much stronger result: in the full MELL fragment two proof-nets R and R' with the same interpretation are the same "up to the connections between the doors of exponential boxes" (we say that they have the same LPS[3] – see Figures 8, 9 and 10 for an example of three different proof-nets having the same LPS). We wrote: "This result can be expressed in terms of differential nets: two cut-free proof-nets with different LPS have different Taylor expansions. We also believe this work is an essential step towards the proof of the full conjecture." Despite the fact we obtained a very interesting result about *all* the proof-nets (i.e. also for non-"connected" proof-nets[4]), the last sentence was a bit too optimistic, since, in the present paper, which presents a proof of the full conjecture, we could not use any previous result nor any previous technique/idea.

The result of the present paper can be seen as

- a semantic separation property in the sense of [11];
- a semantic proof of the confluence property;
- a proof of the "canonicity" of the new syntax of MELL proof-nets;
- a proof of the fact that if the Taylor expansions of two cut-free MELL proof-nets into differential nets[5] [10] coincide, then the two proof-nets coincide.

---

[2] The tradition of the lambda-calculus community rather suggests the word "completeness" and the terminology of category theory rather suggets the word "faithfulness", but we follow here the tradition of the Linear Logic community.

[3] The LPS of a proof-net is the graph obtained by forgetting the outline of the boxes but keeping the auxiliary doors.

[4] and even adding the MIX rule

[5] Differential proof-nets are linear approximations of proof-nets that are meant to allow the expression of the Taylor expansion of proof-nets as infinite series of their linear approximations.

Let us give one more interpretation of its signifance. First, notice that a proof of this result should consist in showing that, given two non β-equivalent proof-nets R and R', their respective semantics $[\![R]\!]$ and $[\![R']\!]$ are not equal, i.e. $[\![R]\!] \setminus [\![R']\!] \neq \emptyset$ or $[\![R']\!] \setminus [\![R]\!] \neq \emptyset$.[6] But, actually, we prove something much stronger: we prove that, given a proof-net R, there exist two points α and β such that, for any proof-net R', we have $\{\alpha, \beta\} \subseteq [\![R']\!] \Leftrightarrow R \simeq_\beta R'$.

Now, the points of the relational model can be seen as non-idempotent intersection types[7] (see [6] and [7] for a correspondance between points of the relational model and System R – System R has also been studied recently in [2]). And the proof given in the present paper uses the types only to derive the normalization property; actually we prove the injectivity for cut-free proof-nets in an untyped framework:[8] substituting the assumption that proof-nets are typed by the assumption that proof-nets are normalizable does not change anything to the proof.[9] In [8], we gave a semantic characterization of normalizable untyped proof-nets and we characterized "head-normalizable" proof-nets as proof-nets having a non-empty interpretation in the relational semantics. Principal typings in untyped λ-calculus are intersection types which allow to recover all the intersection types of some term. If, for instance, we consider the System R of [6] and [7], it is enough to consider some *injective* 1-*point*[10] to obtain the principal typing of an untyped λ-term. But, generally, for normalizable MELL proof-nets, *injective k-points*, for any k, are not principal typings; indeed, two cut-free MELL proof-nets having the same LPS have the same injective k-points for any $k \in \mathbb{N}$. In the current paper we show that a 1-point and a *k-heterogeneous point*[11] together allow to recover the interpretation of any normalizable MELL proof-net. So, the result of the current paper can be seen as a first attempt to find a right notion of "principal typing" of intersection types in Linear Logic. As a consequence, normalization by evaluation, as in [21] for λ-calculus, finally becomes possible in Linear Logic too.

Section 2 formalizes PS's (our cut-free proof-nets). Section 3 gives a sketch of our algorithm leading from $[\![R]\!]$ to the rebuilding of R. Section 4 describes more precisely one step of the algorithm and states our theorem (Theorem 40): $[\![R]\!] = [\![R']\!] \Leftrightarrow R \simeq_\beta R'$, where $\simeq_\beta$ is the reflexive symmetric transitive closure of the cut-elimination relation.

▶ Notations. We denote by ε the empty sequence. If a is a sequence $(\alpha_1, \ldots, \alpha_n)$, then $\alpha_0 : a$ denotes the sequence $(\alpha_0, \ldots, \alpha_n)$; otherwise, it denotes the sequence (α, a) of length 2. The set of finite sequences of elements of some set $\mathcal{E}$ is denoted by $\mathcal{E}^{<\infty}$.

A multiset f of elements of some set $\mathcal{E}$ is a function $\mathcal{E} \to \mathbb{N}$; we denote by $Supp(f)$ *the support of* f i.e. the set $\{e \in \mathcal{E}; f(e) \neq 0\}$. A multiset f is said to be *finite* if $Supp(f)$ is finite. The set of finite multisets of elements of some set $\mathcal{E}$ is denoted by $\mathcal{M}_{fin}(\mathcal{E})$.

---

[6] The converse, i.e. two β-equivalent proof-nets have the same semantics, holds by definition of soundness.

[7] Idempotency of intersection ($\alpha \cap \alpha = \alpha$) does not hold.

[8] For cut-free proof-nets, types guarantee that they are not cyclic as graphs – instead of typing, it is enough to assume this property. Our proof even works for "non-correct" proof-structures (correctness is the property characterizing nets corresponding in a typed framework with proofs in sequent calculus): we could expect that if the injectivity of the relational semantics holds for proof-nets corresponding with MELL sequent calculus, then it still holds for proof-nets corresponding with MELL+MIX sequent calculus, since the category **Rel** of sets and relations is a compact closed category. [13] assuming correctness substituted in the proof the "bridges" of [9] by "empires".

[9] Except that we have to consider the *atomic* subset of the interpretation instead of the full interpretation (see Remark 6).

[10] An *injective k-point* is a point in which all the positive multisets have cardinality k and in which each atom occurring in it occurs exactly twice.

[11] *k-heterogeneous points* are points in which every positive multiset has cardinality k^j for some $j > 0$ and, for any $j > 0$, there is at most one occurrence of a positive multiset having cardinality k^j – they are obtained by *k-heterogeneous experiments* (see our Definition 12). The terminology "k-heterogeneous" has been suggested to us by some reviewer to substitute the expression "k-injective".

If f is a function $\mathcal{E} \to \mathcal{E}'$, $x_0 \in \mathcal{E}$ and $y \in \mathcal{E}'$, then we denote by $f[x_0 \mapsto y]$ the function $\mathcal{E} \to \mathcal{E}'$ defined by $f[x_0 \mapsto y](x) = \begin{cases} f(x) & \text{if } x \neq x_0; \\ y & \text{if } x = x_0. \end{cases}$ If f is a function $\mathcal{E} \to \mathcal{E}'$ and $\mathcal{E}_0 \subseteq \mathsf{dom}(f) = \mathcal{E}$, then we denote by $f[\mathcal{E}_0]$ the set $\{f(x); x \in \mathcal{E}_0\}$.

2 Syntax

We introduce the syntactical objects we are interested in. As recalled in the introduction, simple types guarantee normalization, so we can limit ourselves to nets without any cut. Correctness does not play any role, that is why we do not restrict our nets to be correct and we rather consider proof-structures (*PS's*). Since our proof is easily extended to MELL with axioms, we remove them for simplicity. Moreover, since it is convenient to represent formally our proof using differential nets with boxes (*differential PS's*), we define PS's as differential PS's satisfying some conditions (Definition 4). More generally, *differential* $\circ$*-PS's* are defined by induction on the depth: Definition 1 concerns what happens at depth 0.

We define the set $\mathbb{T}$ of types as follows: $\mathbb{T} ::= 1 \mid \bot \mid (\mathbb{T} \otimes \mathbb{T}) \mid (\mathbb{T} \parr \mathbb{T}) \mid !\mathbb{T} \mid ?\mathbb{T}$. We set $\mathfrak{T} = \{\otimes, \parr, 1, \bot, !, ?, \circ\}$. Pre-contractions ($\circ$-ports) are an artefact of our inductive definition on the depth and are used to ensure the canonicity of our syntactical objects (see Example 6).

▶ **Definition 1.** A *differential ground-structure* is a 6-tuple $\mathcal{G} = (\mathcal{W}, \mathcal{P}, l, t, \mathcal{L}, \mathsf{T})$, where

- $\mathcal{P}$ is a finite set; the elements of $\mathcal{P}(\mathcal{G})$ are the *ports of* $\mathcal{G}$;
- l is a function $\mathcal{P} \to \mathfrak{T}$;the element $l(p)$ of $\mathfrak{T}$ is the *label of* p *in* $\mathcal{G}$;
- $\mathcal{W}$ is a subset of $\{p \in \mathcal{P}; l(p) \neq \circ\}$; the elements of $\mathcal{W}(\mathcal{G})$ are the *wires of* $\mathcal{G}$;
- t is a function $\mathcal{W} \to \{p \in \mathcal{P}; l(p) \notin \{1, \bot\}\}$ such that, for any port p of $\mathcal{G}$, we have $(l(p) \in \{\otimes, \parr\} \Rightarrow \mathsf{Card}(\{w \in \mathcal{W}; t(w) = p\}) = 2)$; if $t(w) = p$, then w is a *premise of* p; the *arity* $a_{\mathcal{G}}(p)$ *of* p is the number of its premises;
- $\mathcal{L}$ is a subset of $\{w \in \mathcal{W}; l(t(w)) \in \{\otimes, \parr\}\}$ such that $(\forall p \in \mathcal{P})\ (l(p) \in \{\otimes, \parr\} \Rightarrow \mathsf{Card}(\{w \in \mathcal{L}; t(w) = p\}) = 1)$; if $w \in \mathcal{L}$ s.t. $t(w) = p$, then w is *the left premise of* p*;*
- and T is a function $\mathcal{P} \to \mathbb{T}$ such that, for any $p \in \mathcal{P}$,
  - if $l(p) \in \{1, \bot\}$, then $\mathsf{T}(p) = l(p)$;
  - if $l(p) = \otimes$ (resp. $l(p) = \parr$), then, for any $w_1 \in \mathcal{W} \cap \mathcal{L}$ and any $w_2 \in \mathcal{W} \setminus \mathcal{L}$ such that $t(w_1) = p = t(w_2)$, we have $\mathsf{T}(p) = (\mathsf{T}(w_1) \otimes \mathsf{T}(w_2))$ (resp. $\mathsf{T}(p) = (\mathsf{T}(w_1) \parr \mathsf{T}(w_2))$);
  - if $l(p) = !$, then $(\exists C \in \mathbb{T})(\mathsf{T}(p) = !C \wedge (\forall w \in \mathcal{W})(t(w) = p \Rightarrow \mathsf{T}(w) = C))$;
  - and if $l(p) \in \{\circ, ?\}$, then $(\exists C \in \mathbb{T})(\mathsf{T}(p) = ?C \wedge (\forall w \in \mathcal{W})(t(w) = p \Rightarrow \mathsf{T}(w) = C))$.

We set $\mathcal{W}(\mathcal{G}) = \mathcal{W}$, $\mathcal{P}(\mathcal{G}) = \mathcal{P}$, $l_{\mathcal{G}} = l$, $t_{\mathcal{G}} = t$, $\mathcal{L}(\mathcal{G}) = \mathcal{L}$, $\mathsf{T}_{\mathcal{G}} = \mathsf{T}$. The set $\mathcal{P}^{\mathsf{f}}(\mathcal{G}) = \mathcal{P} \setminus \mathcal{W}$ is the set of *conclusions of* $\mathcal{G}$. For any $t \in \mathfrak{T}$, we set $\mathcal{P}^t(\mathcal{G}) = \{p \in \mathcal{P}; l(p) = t\}$; we set $\mathcal{P}^m(\mathcal{G}) = \mathcal{P}^{\otimes}(\mathcal{G}) \cup \mathcal{P}^{\parr}(\mathcal{G})$; the set $\mathcal{P}^e(\mathcal{G})$ of *exponential ports of* $\mathcal{G}$ is $\mathcal{P}^!(\mathcal{G}) \cup \mathcal{P}^?(\mathcal{G}) \cup \mathcal{P}^\circ(\mathcal{G})$.

A *ground-structure* is a differential ground-structure $\mathcal{G}$ s.t. $\mathsf{im}(t_{\mathcal{G}}) \cap (\mathcal{P}^!(\mathcal{G}) \cup \mathcal{P}^\circ(\mathcal{G})) = \emptyset$.

Notice that, for any differential ground-structure $\mathcal{G}$, we have $\mathcal{P}^\circ(\mathcal{G}) \subseteq \mathcal{P}^{\mathsf{f}}(\mathcal{G})$.

▶ **Example 2.** The ground-structure $\mathcal{G}$ defined by: $\mathcal{W}(\mathcal{G}) = \{p_3, p_4, p_5\}$; $\mathcal{P}(\mathcal{G}) = \{p_1, \ldots, p_5\}$; $l_{\mathcal{G}}(p_1) = \bot$, $l_{\mathcal{G}}(p_2) = \parr$, $l_{\mathcal{G}}(p_3) = ? = l_{\mathcal{G}}(p_4)$, $l_{\mathcal{G}}(p_5) = 1$;

$t_{\mathcal{G}}(p_3) = p_2 = t_{\mathcal{G}}(p_4)$, $t_{\mathcal{G}}(p_5) = p_4$; $\mathcal{L}(\mathcal{G}) = \{p_3\}$; and $\mathsf{T}_{\mathcal{G}}(p_1) = \bot$, $\mathsf{T}_{\mathcal{G}}(p_2) = (?1 \parr ?1)$, $\mathsf{T}_{\mathcal{G}}(p_3) = ?1 = \mathsf{T}_{\mathcal{G}}(p_4)$, $\mathsf{T}_{\mathcal{G}}(p_5) = 1$; is the ground-structure of the content of the box o_1 of R (the leftmost box of Figure 1).

The content of every box of our *differential* $\circ$*-PS's* is a $\circ$*-PS*: every !-port inside is always the main door of some box.

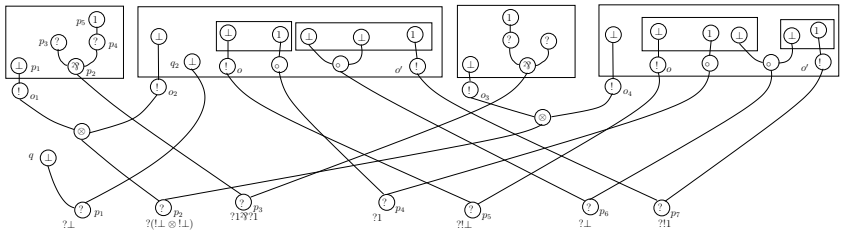


Figure 1 PS R.

▶ **Definition 3.** For any $d \in \mathbb{N}$, we define, by induction on d, the set of *differential* $\circ$*-PS's of depth* d (resp. the set of $\circ$*-PS's of depth* d). A *differential* $\circ$*-PS of depth* d (resp. a $\circ$*-PS of depth* d) is a 4-tuple $S = (\mathcal{G}, \mathcal{B}_0, B, b)$, where

- $\mathcal{G}$ is a differential ground-structure (resp. a ground-structure);
- $\mathcal{B}_0 \subseteq \{p \in \mathcal{P}^!(\mathcal{G}); a_{\mathcal{G}}(p) = 0\}$ (resp. $\mathcal{B}_0 = \mathcal{P}^!(\mathcal{G})$) and is the set of *boxes of* S *at depth* 0;
- B is a function that associates with every $o \in \mathcal{B}_0$ a $\circ$-PS $B(o) = (\mathcal{G}(B(o)), \mathcal{B}_0(B(o)), B_{B(o)}, b_{B(o)})$ of depth $< d$ that enjoys the following property: if $d > 0$, then there exists $o \in \mathcal{B}_0$ s.t. $B(o)$ is a $\circ$-PS of depth $d-1$; the !-port o is *the main door of the box* $B(o)$;
- and b is a function that associates with every $o \in \mathcal{B}_0$ a function $b(o) : \mathcal{P}^{\mathsf{f}}(\mathcal{G}(B(o))) \to \{o\} \cup \mathcal{P}^?(\mathcal{G}) \cup \mathcal{P}^\circ(\mathcal{G})$ such that (resp. $\mathcal{P}^\circ(\mathcal{G}) \subseteq \bigcup_{o \in \mathcal{B}_0} \mathsf{im}(b(o))$ and), for any $o \in \mathcal{B}_0$,
  - $b(o)|_{\mathcal{P}^\circ(\mathcal{G}(B(o)))}$ is injective[12] (resp.[13] $b(o)|_{\mathcal{P}^\circ(\mathcal{G}(B(o)))} = id_{\mathcal{P}^\circ(\mathcal{G}(B(o)))}$); if $q = b(o)(p)$ with $p \in \mathcal{P}^\circ(\mathcal{G}(B(o)))$, then we set $q_{S,o} = p$;
  - $o \in \mathsf{im}(b(o))$ and $b(o)[\mathcal{P}^\circ(\mathcal{G}(B(o)))] \cap \mathcal{P}^!(\mathcal{G}) = \emptyset$;
  - for any $p \in \mathsf{dom}(b(o)) \cap \mathcal{P}^\circ(\mathcal{G}(B_R(o)))$, we have $\mathsf{T}_{\mathcal{G}}(b(o)(p)) = \mathsf{T}_{\mathcal{G}(B(o))}(p)$;
  - for any $p \in \mathsf{dom}(b(o)) \backslash \mathcal{P}^\circ(\mathcal{G}(B_R(o)))$, we have $\mathsf{T}_{\mathcal{G}}(b(o)(p)) \in \{?\mathsf{T}_{\mathcal{G}(B(o))}(p), !\mathsf{T}_{\mathcal{G}(B(o))}(p)\}$;

(resp. moreover no $p \in \mathcal{P}(\mathcal{G})$ is a sequence)[14]. For any differential $\circ$-PS $S = (\mathcal{G}, \mathcal{B}_0, B, b)$, we set $\mathcal{G}(S) = \mathcal{G}$, $\mathcal{B}_0(S) = \mathcal{B}_0$ and $\mathcal{B}(S) = \mathcal{B}_0(S) \cup \bigcup_{o \in \mathcal{B}_0(S)} \{o : o'; o' \in \mathcal{B}(B_S(o))\}$ is the set of *boxes of* S. We denote by B_S the extension of the function B that associates with each $o : o' \in \mathcal{B}(S)$, where $o \in \mathcal{B}_0(S)$, the $\circ$-PS $B_{B_S(o)}(o')$. We denote by b_S the extension of the function b that associates with each $o : o' \in \mathcal{B}(S)$, where $o \in \mathcal{B}_0(S)$, the function $b_{B_S(o)}(o')$. We set $\mathcal{W}_0(S) = \mathcal{W}(\mathcal{G}(S))$ and $\mathcal{P}_0(S) = \mathcal{P}(\mathcal{G}(S))$; the elements of $\mathcal{P}_0(S)$ (resp. of $\mathcal{W}_0(S)$) are the *ports of* S *at depth* 0 (resp. the *wires of* S *at depth* 0). For any $l \in \mathfrak{T} \cup \{m, e\}$, we set $\mathcal{P}_0^l(S) = \mathcal{P}^l(\mathcal{G}(S))$. We set $\mathcal{P}^{\mathsf{f}}(S) = \mathcal{P}^{\mathsf{f}}(\mathcal{G}(S))$, $\mathcal{P}^{\mathsf{f}}_\circ(S) = \mathcal{P}^\circ(\mathcal{G}(S))$ and $\mathcal{P}^{\mathsf{f}}_\bullet(S) = \mathcal{P}^{\mathsf{f}}(S) \setminus \mathcal{P}^{\mathsf{f}}_\circ(S)$; the elements of $\mathcal{P}^{\mathsf{f}}(S)$ are the *conclusions of* S and the elements of $\mathcal{P}^{\mathsf{f}}_\circ(S)$ are the $\circ$*-conclusions of* S. For any relation $P \in \{\geq, =, <\}$ on $\mathbb{N}$, for any $i \in \mathbb{N}$, we set $\mathcal{B}_0^{Pi}(S) = \{o \in \mathcal{B}_0(S); depth(B_S(o))Pi\}$ and $\mathcal{B}^{Pi}(S) = \{o \in \mathcal{B}(S); depth(B_S(o))Pi\}$.

PS's are the MELL proof-nets studied in the present paper: there is no cut and no assumption of correctness property.

▶ **Definition 4.** A PS is a $\circ$-PS R such that $\mathcal{P}^{\mathsf{f}}_\circ(R) = \emptyset$.

▶ **Example 5.** Consider the PS R of Figure 1. We have $\mathcal{B}_0(R) = \{o_1, o_2, o_3, o_4\}$, $\mathcal{B}(R) = \{o_1, o_2, o_3, o_4, (o_2, o), (o_2, o'), (o_4, o), (o_4, o')\}$, $\mathcal{B}^{=0}(R) = \{o_1, (o_2, o), (o_2, o'), o_3, (o_4, o), (o_4, o')\}$. We have $b_R(o_2)(o) = p_5$, $b_R(o_2)(p_4) = p_4$, $b_R(o_2)(p_6) = p_6$ and $b_R(o_2)(o') = p_7$.

[12] So one cannot (pre-)contract several $\circ$-ports of the same box.
[13] This stronger condition on $\circ$-PS's is *ad hoc*, but it allows to lighten the notations.
[14] This condition on $\circ$-PS's is *ad hoc*, but it allows to simplify Definition 14.

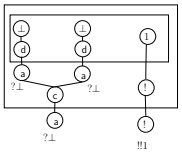


Figure 2 O_1 ("old syntax").

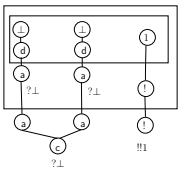


Figure 3 O_2 ("old syntax").

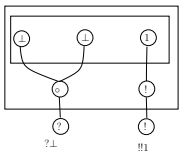


Figure 4 PS N.

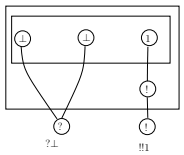


Figure 5 Danos-Regnier-proof-net obtained from the PS N by "ignoring" the $\circ$-ports.

▶ **Example 6.** In order to understand the role of the $\circ$-ports, consider how the proof-nets O_1 (Figure 2) and O_2 (Figure 3) in the "old syntax" (we denoted derelictions, contractions and auxiliary doors of the "old syntax" by d, c and a, respectively) are represented by the same PS N (Figure 4). Roughly speaking, in our formalism, one pre-contracts (using $\circ$-ports) as soon as possible and one contracts (using ?-ports) as late as possible. Notice that if one "ignores" the $\circ$-ports, i.e. if, whenever a port p that is not a $\circ$-port is immediately above a series of $\circ$-ports that are imediately above a contraction q, one draws a wire from p to q and one removes all the $\circ$-ports, then one obtains a cut-free proof-net of [4] without the "sequentialization condition" (see Figure 5); and, conversely, given such a proof-net, there is a unique way to add the $\circ$-ports to obtain our PS's. So our definition of PS's is exactly equivalent to the definition of cut-free proof-nets of [4] without the "sequentialization condition"; one reason to not take the same definition as the one of [4] is the desire to have an inductive definition on the depth, which, as a consequence, leads to the auxiliary notion of $\circ$-PS.

We write $R \simeq R'$ (resp. $R \equiv R'$) if R and R' are the same differential PS's up to the names of their ports (resp. that are not conclusions):

▶ **Definition 7.** An *isomorphism* $\varphi : \mathcal{G} \simeq \mathcal{G}'$ *of ground-structures* is a structure-preserving bijection $\mathcal{P}_0(\mathcal{G}) \simeq \mathcal{P}_0(\mathcal{G}')$. We define, by induction on *depth*(R), when $\varphi : R \simeq R'$ holds for two differential $\circ$-PS's R and R': it holds whenever φ is a pair $(\varphi_{\mathcal{G}}, (\varphi_o)_{o \in \mathcal{B}_0(R)})$ s.t. $\varphi_{\mathcal{G}} : \mathcal{G}(R) \simeq \mathcal{G}(R')$, $\mathcal{B}_0(R') = \varphi_{\mathcal{G}}[\mathcal{P}_0^!(R)]$ and, for any $o \in \mathcal{B}_0(R)$, $\varphi_o : B_R(o) \simeq B_{R'}(\varphi_{\mathcal{G}}(o))$ and $(\forall q \in \mathcal{P}^{\mathsf{f}}(B_R(o)))b_{R'}(\varphi_{\mathcal{G}}(o))(\mathcal{G}(\varphi_o)(q)) = \varphi_{\mathcal{G}}(b_R(o)(q))$. We set $\mathcal{G}(\varphi) = \varphi_{\mathcal{G}}$ and, for any $o \in \mathcal{B}_0(R)$, $\varphi(o) = \varphi_o$. We write $\varphi : R \equiv R'$ if $\varphi : R \simeq R'$ s.t. $\mathcal{G}(\varphi)|_{\mathcal{P}^{\mathsf{f}}(R)} = id_{\mathcal{P}^{\mathsf{f}}(R)}$. We write $R \simeq R'$ (resp. $R \equiv R'$) if there exists φ s.t. $\varphi : R \simeq R'$ (resp. $\varphi : R \equiv R'$).

The *arity* $a_R(q)$ *of a port* q *in a differential* $\circ$*-PS* R is computed by "ignoring" the $\circ$-conclusions of the boxes of R:

▶ **Definition 8.** Let R be a differential $\circ$-PS. We define, by induction on *depth*(R), the integers $a_R(q)$ for any $q \in \mathcal{P}_0(R)$ and *cosize*(R): we set $a_R(q) = a_{\mathcal{G}(R)}(q) + \sum_{o \in \mathcal{B}_0(R)} \mathsf{Card}\left(\left\{p \in \mathcal{P}_\bullet^{\mathsf{f}}(B_R(o)); b_R(o)(p) = q\right\}\right) + \sum_{\substack{o \in \mathcal{B}_0(R) \\ q \in b_R(o)[\mathcal{P}_\circ^{\mathsf{f}}(B_R(o))]}} a_{B_R(o)}(q_{R,o})$ and *cosize*$(R) = \max(\{a_R(p); p \in \mathcal{P}_0(R)\} \cup \{\mathit{cosize}(B_R(o)); o \in \mathcal{B}_0(R)\})$.

▶ **Example 9.** We have $a_R(p_6) = 4$ (and not 2) and *cosize*$(R) = 4$ (see Figure 1).

3 Experiments and their partial expansions

When Jean-Yves Girard introduced proof-nets in [12], he also introduced *experiments of proof-nets*. Experiments (see our Definition 10) are a technology allowing to compute pointwise the

interpretation $[\![R]\!]$ of a proof-net R in the model directly on the proof-net rather than through some sequent calculus proof obtained from one of its sequentializations: the set of *results* of all the experiments of a given proof-net is its interpretation $[\![R]\!]$. In an untyped framework, experiments correspond with type derivations and results correspond with intersection types.

▶ **Definition 10.** For any $C \in \mathbb{T}$, we define, by induction on C, the set $[\![C]\!]$: $[\![1]\!] = \{*\} = [\![\bot]\!]$; $[\![(C_1 \otimes C_2)]\!] = [\![C_1]\!] \times [\![C_2]\!] = [\![(C_1 \parr C_2)]\!]$; $[\![!C]\!] = \mathcal{M}_{fin}([\![C]\!]) = [\![?C]\!]$.

Let R be a differential $\circ$-PS. We define, by induction on $depth(R)$, the set of *experiments of* R: it is is the set of triples $(R, e_{\mathcal{P}}, e_{\mathcal{B}})$, where $e_{\mathcal{P}}$ is a function that associates with every $p \in \mathcal{P}_0(R)$ an element of $[\![\mathsf{T}_{\mathcal{G}(R)}(p)]\!]$ and $e_{\mathcal{B}}$ is a function which associates to every $o \in \mathcal{B}_0(R)$ a finite multiset of experiments of $B_R(o)$ such that

- for any $p \in \mathcal{P}_0^m(R)$, for any $w_1, w_2 \in \mathcal{W}_0(R)$ such that $t_{\mathcal{G}(R)}(w_1) = p = t_{\mathcal{G}(R)}(w_2)$, $w_1 \in \mathcal{L}(\mathcal{G}(R))$ and $w_2 \notin \mathcal{L}(\mathcal{G}(R))$, we have $e_{\mathcal{P}}(p) = (e_{\mathcal{P}}(w_1), e_{\mathcal{P}}(w_2))$;
- for any $p \in \mathcal{P}_0^e(R)$, we have $e(p) = \sum_{\substack{w \in \mathcal{W}_0(R) \\ t_{\mathcal{G}(R)}(w)=p}} [e_{\mathcal{P}}(w)] + \sum_{o \in \mathcal{B}_0(R)} \sum_{e' \in Supp(e_{\mathcal{B}}(o))} (\sum_{\substack{q \in \mathcal{P}^{\mathsf{f}}_{\bullet}(B_R(o)) \\ b_R(o)(q)=p}} e_{\mathcal{B}}(o)(e') \cdot [e'_{\mathcal{P}}(q)] + \sum_{\substack{q \in \mathcal{P}^{\mathsf{f}}_{\circ}(B_R(o)) \\ b_R(o)(q)=p}} e_{\mathcal{B}}(o)(e') \cdot e'_{\mathcal{P}}(q))$.

For any experiment $e = (R, e_{\mathcal{P}}, e_{\mathcal{B}})$, we set $\mathcal{P}(e) = e_{\mathcal{P}}$ and $\mathcal{B}(e) = e_{\mathcal{B}}$. We set $[\![R]\!] = \{\mathcal{P}(e)|_{\mathcal{P}^{\mathsf{f}}(R)}; e \text{ is an experiment of } R\}$.

We encode in a more compact way the "relevant" information given by an experiment *via pseudo-experiments* and the functions $e^{\#}$:

▶ **Definition 11.** For any differential $\circ$-PS R, we define, by induction on $depth(R)$, the set of *pseudo-experiments of* R: it is the set of functions that associate with every $o \in \mathcal{B}_0(R)$ a finite set of pseudo-experiments of $B_R(o)$ and with ε a pair (R, m) for some $m \in \mathbb{N}$.

Given an experiment e of some differential $\circ$-PS R, we define, by induction on $depth(R)$, a pseudo-experiment $\overline{e}$ of R as follows: $\overline{e}(\varepsilon) = (R, 1)$ and $\overline{e}(o) = \bigcup_{f \in Supp(\mathcal{B}(e)(o))} \{\overline{f}[\varepsilon \mapsto (B_R(o), i)]; 1 \leq i \leq \mathcal{B}(e)(o)(f)\}$ for any $o \in \mathcal{B}_0(R)$.

Given a pseudo-experiment e of a differential $\circ$-PS R, we define, by induction on $depth(R)$, the function $e^{\#} : \mathcal{B}(R) \to \mathcal{P}_{fin}(\mathbb{N})$ as follows: for any $o \in \mathcal{B}_0(R)$, $e^{\#}(o) = \{\mathsf{Card}\,(e(o))\}$ and, for any $o' \in \mathcal{B}(B_R(o))$, $e^{\#}(o : o') = \bigcup_{e' \in e(o)} e'^{\#}(o')$.

There are different kinds of experiments:

- In [24], it was shown that given the result of an *injective k-obsessional experiment* (k big enough) of a cut-free proof-net in the fragment $A ::= X|?A \parr A|A \parr ?A|A \otimes A|!A$, one can rebuild the entire experiment and, so, the entire proof-net. There, "injective" means that the experiment labels two different axioms with different atoms and "obsessional" means that different copies of the same axiom are labeled by the same atom.
- In [9], it was shown that for any two cut-free MELL proof-nets R and R', we have $LPS(R) = LPS(R')$ iff, for k big enough[15], there exist an *injective k-experiment* of R and an *injective k-experiment* of R' having the same result; as an immediate corollary we obtained the injectivity of the set of (recursively) connected proof-nets. There, "injective" means that not only the experiment labels two different axioms with different atoms, but it labels also different copies of the same axiom by different atoms. Given some proof-net R, there is exactly one injective k-experiment of R up to the names of the atoms.

[15] Interestingly, [13], following the approach of [9], showed that, if these two proof-nets are assumed to be (recursively) connected, then we can take $k = 2$.

- In the present paper we show that, for any two PS's R and R', given the result α of a *k-heterogeneous experiment* of R for k big enough, if $\alpha \in [\![R']\!]$, then R' is the same PS as R. The conditions on k are given by the result of a 1-experiment, so we show that two (well-chosen) points are enough to determine a PS. The expression "k-heterogeneous" means that, for any two different occurrences of boxes, the experiment never takes the same number of copies: it takes k^{j_1} copies and k^{j_2} copies with $j_1 \neq j_2$ (*a contrario*, in [24] and [9], the experiments always take the same number of copies). As shown by the proof-net S of Figure 11, it is impossible to rebuild the experiment from its result, since there exist four different 4-heterogeneous experiments e_1, e_2, e_3 and e_4 such that, for any $i \in \{1,2,3,4\}$, we have $e_i(p) = (*,*)$, $e_i(o_1) = [*,*,*,*]$ and $e_i(p') = [[\underbrace{*,\dots,*}_{4^2}],\dots,[\underbrace{*,\dots,*}_{4^6}]]$. For instance e_1 takes 4 copies of the box o_1 and 16 copies of the box o_2, while e_2 takes 4 copies of the box o_1 and 64 copies of the box o_2.

▶ **Definition 12.** Let $k > 1$. A pseudo-experiment e of a $\circ$-PS R is said to be *k-heterogeneous* if

- for any $o \in \mathcal{B}(R)$, for any $m \in e^{\#}(o)$, there exists $j > 0$ such that $m = k^j$;
- for any $o \in \mathcal{B}_0(R)$, for any $o' \in \mathcal{B}(B_R(o))$, we have $(\forall e_1, e_2 \in e(o))\ ({e_1}^{\#}(o') \cap {e_2}^{\#}(o') \neq \emptyset \Rightarrow e_1 = e_2)$;
- and, for any $o_1, o_2 \in \mathcal{B}(R)$, we have $(e^{\#}(o_1) \cap e^{\#}(o_2) \neq \emptyset \Rightarrow o_1 = o_2)$.

An experiment e is said to be *k-heterogeneous* if $\overline{e}$ is k-heterogeneous.

▶ **Example 13.** There exists a 10-heterogeneous pseudo-experiment f of the proof-net R of Figure 1 such that $f^{\#}(o_1) = \{10^{223}\}$, $f^{\#}(o_2) = \{10\}$, $f^{\#}(o_3) = \{10^{224}\}$, $f^{\#}(o_4) = \{100\}$, $f^{\#}((o_2, o)) = \{10^3, \dots, 10^{12}\}$, $f^{\#}((o_2, o')) = \{10^{13}, \dots, 10^{22}\}$, $f^{\#}((o_4, o)) = \{10^{23}, \dots, 10^{122}\}$ and $f^{\#}((o_4, o')) = \{10^{123}, \dots, 10^{222}\}$.

In [9], the interest for *injective* experiments came from the remark that the result of an *injective* experiment of a *cut-free* proof-net can be easily identified with a differential net of its Taylor expansion in a sum of differential nets [10] (it is essentialy the content of our Lemma 16). Thus any proof using injective experiments can be straightforwardly expressed in terms of differential nets and conversely. Since this identification is trivial, besides the idea of considering injective experiments instead of obsessional experiments, the use of the terminology of differential nets does not bring any new insight[16], it just superficially changes the presentation.That is why we decided in [9] to avoid introducing explicitely differential nets. In the present paper, we made the opposite choice for the following reason: the algorithm leading from the result of a k-heterogeneous experiment of R to the entire rebuilding of R is done in several steps: in the intermediate steps, we obtain a partial rebuilding where some boxes have been recovered but not all of them; a convenient way to represent this information is the use of "differential nets with boxes" (called "differential PS's" in the present paper) that lie between the purely linear differential proof-nets and the non-linear proof-nets. Now, the differential net representing the result and the proof-net R are both instances of the more general notion of "differential nets with boxes".

The rebuilding of the proof-net R is done in d steps, where d is the depth of R. We first rebuild the occurrences of the boxes of depth 0 (the deepest ones) and next we rebuild the

[16] For proof-nets with cuts, the situation is completely different: the great novelty of differential nets is that differential nets have a cut-elimination; the differential nets appearing in the Taylor expansion of a proof-net with cuts have cuts, while the semantics does not see these cuts. But the proofs of the injectivity only consider cut-free proof-nets.

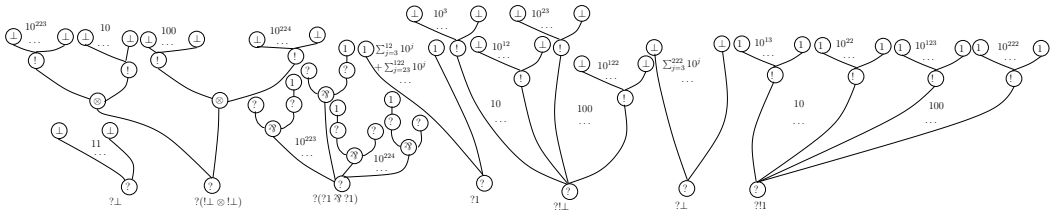


Figure 6 $\mathcal{T}(f)[0]$.

occurrences of the boxes of depth 1 and so on... This can be formalized using differential nets (with boxes) as follows: if e is an injective experiment of R, then $\mathcal{T}(\overline{e})[i]$ is the differential net corresponding with e in which only boxes of depth $\geq i$ are expanded,[17] so $\mathcal{T}(\overline{e})[0]$ is (essentially) the same as the result of the experiment and $\mathcal{T}(\overline{e})[d] = R$; the first step of the algorithm builds $\mathcal{T}(\overline{e})[1]$ from $\mathcal{T}(\overline{e})[0]$, the second step builds $\mathcal{T}(\overline{e})[2]$ from $\mathcal{T}(\overline{e})[1]$, and so on... We thus reduced the problem of the injectivity to the problem of rebuilding $\mathcal{T}(\overline{e})[i+1]$ from $\mathcal{T}(\overline{e})[i]$ for any k-heterogeneous experiment e (k big enough).

▶ **Definition 14.** Let R be a $\circ$-PS of depth d. Let e be a pseudo-experiment of R. Let $i \in \mathbb{N}$. We define, by induction on d, a differential $\circ$-PS $\mathcal{T}(e)[i] = ((\mathcal{W}_{e,i}, \mathcal{P}_{e,i}, l_{e,i}, t_{e,i}, \mathcal{L}_{e,i}, \mathsf{T}_{e,i}), \mathcal{B}_{e,i}, B_{e,i}, b_{e,i})$ of depth $\min\{i, d\}$ s.t. $\mathcal{P}^{\mathsf{f}}(R) = \mathcal{P}^{\mathsf{f}}(\mathcal{T}(e)[i])$ and $(\forall p \in \mathcal{P}^{\mathsf{f}}(R)) l_{\mathcal{G}(R)}(p) = l_{\mathcal{G}(\mathcal{T}(e)[i])}(p)$ as follows: we set $\mathcal{P}^{\bullet}_{e,i} = \bigcup_{o_1 \in \mathcal{B}_0^{\geq i}(R)} \bigcup_{e_1 \in e(o_1)} \left\{(o_1, e_1) : p; p \in \mathcal{P}_{e_1,i} \setminus \mathcal{P}^{\mathsf{f}}_{\circ}(B_R(o_1))\right\}$;

- $\mathcal{W}_{e,i} = \mathcal{W}_0(R) \cup \mathcal{P}^{\bullet}_{e,i}$ and $\mathcal{P}_{e,i} = \mathcal{P}_0(R) \cup \mathcal{P}^{\bullet}_{e,i}$;
- $l_{e,i}(p) = \begin{cases} l_{\mathcal{G}(R)}(p) & \text{if } p \in \mathcal{P}_0(R); \\ l_{e_1,i}(p') & \text{if } p = (o_1, e_1) : p' \text{ with } o_1 \in \mathcal{B}_0^{\geq i}(R); \end{cases}$
- $t_{e,i}$ is the extension of $t_{\mathcal{G}(R)}$ that associates with each $(o_1, e_1) : w' \in \mathcal{W}_{e,i}$, where $o_1 \in \mathcal{B}_0^{\geq i}(R)$, the port $\begin{cases} (o_1, e_1) : t_{e_1,i}(w') & \text{if } w' \in \mathcal{W}_{e_1,i} \text{ and } t_{e_1,i}(w') \notin \mathcal{P}^{f}_{\circ}(B_R(o_1)); \\ b_R(o_1)(t_{e_1,i}(w')) & \text{if } w' \in \mathcal{W}_{e_1,i} \text{ and } t_{e_1,i}(w') \in \mathcal{P}^{f}_{\circ}(B_R(o_1)); \\ b_R(o_1)(w') & \text{if } w' \in \mathcal{P}^{f}_{\bullet}(B_R(o_1)); \end{cases}$
- $\mathcal{L}_{e,i} = \mathcal{L}(\mathcal{G}(R)) \cup \bigcup_{o_1 \in \mathcal{B}_0^{\geq i}(R)} \bigcup_{e_1 \in e(o_1)} \{(o_1, e_1) : p; p \in \mathcal{L}_{e_1,i}\}$
- $\mathsf{T}_{e,i}(p) = \begin{cases} \mathsf{T}_{\mathcal{G}(R)}(p) & \text{if } p \in \mathcal{P}_0(R); \\ \mathsf{T}_{e_1,i}(p') & \text{if } p = (o_1, e_1) : p' \text{ with } o_1 \in \mathcal{B}_0^{\geq i}(R); \end{cases}$
- $\mathcal{B}_{e,i} = \mathcal{B}_0^{<i}(R) \cup \bigcup_{o_1 \in \mathcal{B}_0^{\geq i}(R)} \bigcup_{e_1 \in e(o_1)} \{\ (o_1, e_1) : o'\ ;\ o' \in \mathcal{B}_{e_1,i}\}$
- $B_{e,i}(o) = \begin{cases} B_R(o) & \text{if } o \in \mathcal{B}_0^{<i}(R); \\ B_{e_1,i}(o') & \text{if } o = (o_1, e_1) : o' \text{ with } o_1 \in \mathcal{B}_0^{\geq i}(R); \end{cases}$
- $b_{e,i}$ is the extension of $b_{R|\mathcal{B}_0^{<i}(R)}$ that associates with each $(o_1, e_1) : o' \in \mathcal{B}_{e,i}$, where $o_1 \in \mathcal{B}_0^{\geq i}(R)$, the function $p \mapsto \begin{cases} (o_1, e_1) : b_{e_1,i}(o')(p) & \text{if } b_{e_1,i}(o')(p) \notin \mathcal{P}^{f}_{\circ}(B_R(o_1)); \\ b_{e_1,i}(o')(p) & \text{if } b_{e_1,i}(o')(p) \in \mathcal{P}^{f}_{\circ}(B_R(o_1)). \end{cases}$

▶ **Example 15.** If f is a pseudo-experiment of the proof-net R of Figure 1 with $f^{\#}$ like in Example 13, then Figures 6 and 7 represent respectively $\mathcal{T}(f)[0]$ and $\mathcal{T}(f)[1]$.

The injectivity of the relational semantics for differential PS's of depth 0 is trivial (one can proceed by induction on the cardinality of the set of ports). Since $[\![\mathcal{T}(\overline{e})[0]]\!] = \{e_{|\mathcal{P}^{\mathsf{f}}(R)}\}$, one can easily identify the result $e_{|\mathcal{P}^{\mathsf{f}}(R)}$ of an experiment e with the differential net $\mathcal{T}(\overline{e})[0]$:

[17] Boxes *of* depth $\geq i$ are boxes whose content is a proof-net of depth $\geq i$; the reader should not confuse *boxes of depth* $\geq i$ with *boxes at depth* $\geq i$.

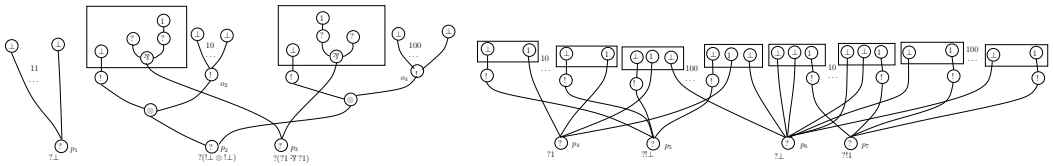


Figure 7 $\mathcal{T}(f)[1]$.

▶ **Lemma 16.** *Let R and R' be two $\circ$-PS's such that $\mathcal{P}^f(R) = \mathcal{P}^f(R')$. Let e be an experiment of R and let e' be an experiment of R' such that $e_{|\mathcal{P}^f(R)} = e'_{|\mathcal{P}^f(R')}$. Then $\mathcal{T}(\overline{e})[0] \equiv \mathcal{T}(\overline{e'})[0]$.*

Now, the following fact shows that if we are able to recover $\mathcal{T}(\overline{e})[depth(R)]$ from $\mathcal{T}(\overline{e})[0]$, then we are done.

▶ **Fact 17.** *Let R be a $\circ$-PS. Let e be a pseudo-experiment of R. Then $\mathcal{T}(e)[depth(R)] = R$.*

If e is a k-heterogeneous experiment of R, then, for any $i \in \mathbb{N}$, there exists a bijection $!_{e,i} : \bigcup_{o\in\mathcal{B}^{\geq i}(R)}\{\log_k(m); m \in e^{\#}(o)\} \simeq \mathcal{P}^!_0(\mathcal{T}(e)[i]) \setminus \mathcal{B}_0(\mathcal{T}(e)[i])$ such that, for any $j \in \mathsf{dom}(!_{e,i})$, we have $(a_{\mathcal{T}(e)[i]} \circ !_{e,i})(j) = k^j$. In Subsection 4.1, we will show how to recover $\bigcup_{o\in\mathcal{B}^{\geq i}(R)}\{\log_k(m); m \in e^{\#}(o)\}$ from $\mathcal{T}(e)[0]$. There are two kinds of boxes of $\mathcal{T}(e)[i+1]$ at depth 0: the "new" boxes of depth i and the boxes of depth $< i$, which are the "old" boxes (i.e. that already were in $\mathcal{T}(e)[i]$) that do not go inside some "new" box:

▶ **Fact 18.** *Let R be a $\circ$-PS. Let e be a pseudo-experiment of R. Let $i \in \mathbb{N}$. Then we have*

- $\mathcal{B}_0^{<i}(\mathcal{T}(e)[i+1]) = \mathcal{B}_0(\mathcal{T}(e)[i]) \cap \mathcal{P}^!_0(\mathcal{T}(e)[i+1])$;
- $B_{\mathcal{T}(e)[i+1]}|_{\mathcal{B}_0^{<i}(\mathcal{T}(e)[i+1])} = B_{\mathcal{T}(e)[i]}|_{\mathcal{B}_0^{<i}(\mathcal{T}(e)[i+1])}$;
- *and* $b_{\mathcal{T}(e)[i+1]}|_{\mathcal{B}_0^{<i}(\mathcal{T}(e)[i+1])} = b_{\mathcal{T}(e)[i]}|_{\mathcal{B}_0^{<i}(\mathcal{T}(e)[i+1])}$.

The challenge is the rebuilding of the "new" boxes at depth 0 of depth i.

4 From $\mathcal{T}(e)[i]$ to $\mathcal{T}(e)[i+1]$

4.1 The outline of the boxes

In this subsection we first show how to recover the set $\bigcup_{o\in\mathcal{B}^{\geq i}(R)}\{\log_k(m); m \in e^{\#}(o)\}$ and, therefore, the set $\mathcal{P}^!_0(\mathcal{T}(e)[i]) \setminus \mathcal{B}_0(\mathcal{T}(e)[i])$ (Lemma 21). Next, we show how to determine, from $\mathcal{T}(e)[i]$, the set $\mathcal{B}_0^{=i}(\mathcal{T}(e)[i+1])$ of "new" boxes and, for any such "new" box $o \in \mathcal{B}_0^{=i}(\mathcal{T}(e)[i+1])$, the set $\mathsf{im}(b_{\mathcal{T}(e)[i+1]}(o))$ of exponential ports that are immediately below (Proposition 25). In particular, we have $\mathcal{B}_0^{=i}(\mathcal{T}(e)[i+1]) =!_{e,i}[\mathcal{N}_i(e)]$, where the set $\mathcal{N}_i(e) \subseteq \mathbb{N}$ is defined from the set $\mathcal{M}_0(e)$ of the numbers of copies of boxes taken by the pseudo–experiment e:

▶ **Definition 19.** Let R be a differential $\circ$-PS. Let $k > 1$. Let e be a k-heterogeneous pseudo-experiment of R. For any $i \in \mathbb{N}$, we define, by induction on i, $\mathcal{M}_i(e) \subseteq \mathbb{N} \setminus \{0\}$ and $(m_{i,j}(e))_{j\in\mathbb{N}} \in \{0, \dots, k-1\}^{\mathbb{N}}$ as follows. We set $\mathcal{M}_0(e) = \bigcup_{o\in\mathcal{B}(R)}\{j \in \mathbb{N}; k^j \in e^{\#}(o)\}$ and we write $\mathsf{Card}\,(\mathcal{M}_i(e))$ in base k: $\mathsf{Card}\,(\mathcal{M}_i(e)) = \sum_{j\in\mathbb{N}} m_{i,j}(e) \cdot k^j$; we set $\mathcal{M}_{i+1}(e) = \{j > 0; m_{i,j}(e) \neq 0\}$.

For any $i \in \mathbb{N}$, we set $\mathcal{N}_i(e) = \mathcal{M}_i(e) \setminus \mathcal{M}_{i+1}(e)$.

Notice that all the sets $\mathcal{M}_i(e)$ and $\mathcal{N}_i(e)$ can be computed from $\mathcal{T}(e)[0]$, since we have $\mathcal{M}_0(e) = \{a_{\mathcal{T}(e)[0]}(p); p \in \mathcal{P}^!_0(\mathcal{T}(e)[0])\}$.

▶ **Example 20.** If f is a 10-heterogeneous pseudo-experiment as in Example 13, then $\mathcal{M}_0(f) = \{1, \dots, 224\}$. We have $\mathsf{Card}(\mathcal{M}_0(f)) = 4 + 2 \cdot 10^1 + 2 \cdot 10^2$, hence $\mathcal{M}_1(f) = \{1, 2\}$ and $\mathcal{N}_0(f) = \{3, \dots, 224\}$. We have $\mathsf{Card}(\mathcal{M}_1(f)) = 2$, hence $\mathcal{M}_2(f) = \emptyset$ and $\mathcal{N}_1(f) = \{1, 2\}$.

The following lemma shows that, for any k-heterogeneous pseudo-experiment e of R, for any $i \in \mathbb{N}$, the function $!_{e,i}$ is actually a bijection $\mathcal{M}_i(e) \to \mathcal{P}_0^!(\mathcal{T}(e)[i]) \setminus \mathcal{B}_0(\mathcal{T}(e)[i])$ such that, for any $j \in \mathcal{M}_i(e)$, we have $(a_{\mathcal{T}(e)[i]} \circ !_{\mathcal{T}(e)[i]})(j) = k^j$.

▶ **Lemma 21.** *Let R be a $\circ$-PS. Let $k > \mathsf{Card}(\mathcal{B}(R))$. For any k-heterogeneous pseudo-experiment e of R, for any $i \in \mathbb{N}$, we have $\mathcal{M}_i(e) = \bigcup_{o \in \mathcal{B}^{\geq i}(R)} \{j \in \mathbb{N}; k^j \in e^{\#}(o)\}$, hence $\mathcal{N}_i(e) = \bigcup_{o \in \mathcal{B}^{=i}(R)} \{j \in \mathbb{N}; k^j \in e^{\#}(o)\}$.*

▶ **Example 22** (Continuation of Example 20)**.** We thus have $\mathcal{M}_1(f) = \{1, 2\}$ and $\mathcal{P}_0^!(\mathcal{T}(f)[i]) \setminus \mathcal{B}_0(\mathcal{T}(f)[1]) = \{o_2, o_4\}$ with $a_{\mathcal{T}(f)[1]}(o_2) = 10^1$ and $a_{\mathcal{T}(f)[1]}(o_4) = 10^2$ (see Figure 7).

The set $\mathcal{K}_{k,\mathcal{N}_i(e)}(S)$ of "critical ports" is a set of exponential ports that will play a crucial role in our algorithm.

▶ **Definition 23.** Let S be a differential $\circ$-PS. Let $k > 1$. For any $p \in \mathcal{P}_0(S)$, we define the sequence $(m_{k,j}(S)(p))_{j \in \mathbb{N}} \in \{0, \dots, k-1\}^{\mathbb{N}}$ as follows: $a_S(p) = \sum_{j \in \mathbb{N}} m_{k,j}(S)(p) \cdot k^j$. For any $j \in \mathbb{N}$, we set $\mathcal{K}_{k,j}(S) = \{p \in \mathcal{P}_0(S); m_{k,j}(S)(p) \neq 0\} \cap \mathcal{P}^e(\mathcal{G}(S))$ and, for any $J \subseteq \mathbb{N}$, we set $\mathcal{K}_{k,J}(S) = \bigcup_{j \in J} \mathcal{K}_{k,j}(S)$.

▶ **Example 24.** We have $\mathcal{K}_{10,1}(S) = \{p_1, p_4, p_5, p_6, p_7, o_2\}$ and $\mathcal{K}_{10,2}(S) = \{p_4, p_5, p_6, p_7, o_4\}$, where S is the PS of Figure 7. So we have $\mathcal{K}_{10,\{1,2\}}(S) = \{p_1, p_4, p_5, p_6, p_7, o_2, o_4\}$.

Critical ports are defined by their arities. We show that they are exponential ports that are immediately below the "new" boxes:

▶ **Proposition 25.** *Let R be a $\circ$-PS. Let $k > \mathsf{Card}(\mathcal{B}(R)), \mathit{cosize}(R)$. Let e be a k-heterogeneous pseudo-experiment of R and let $i \in \mathbb{N}$. Then we have $\mathcal{B}_0^{=i}(\mathcal{T}(e)[i+1]) = !_{e,i}[\mathcal{N}_{i,e}]$. Furthermore, for any $j \in \mathcal{N}_i(e)$, we have $\mathsf{im}(b_{\mathcal{T}(e)[i+1]}(!_{e,i}(j))) = \mathcal{K}_{k,j}(\mathcal{T}(e)[i])$ and, if $!_{e,i}(j) \notin \mathcal{B}_0^{=i}(R)$, then there exist $o_1 \in \mathcal{B}_0^{\geq i+1}(R)$ and $e_1 \in e(o_1)$ such that $j \in \mathcal{N}_i(e_1)$. In particular, we have $\mathcal{K}_{k,\mathcal{N}_i(e)}(\mathcal{T}(e)[i]) \subseteq \mathcal{P}_0^e(\mathcal{T}(e)[i+1])$.*

In particular, this proposition highlights one more essential difference between the k-experiments of [23, 24, 9, 13] and our k-heterogeneous experiments. There, such a k-experiment labelling some contraction p with a multiset of cardinality $\sum_j m_j \cdot k^j$ (where $0 \leq m_j < k$ for any j) gives the information that immediately above the contraction p there are exactly m_{j_0} series of exactly j_0 auxiliary doors. Here, whenever a k-heterogeneous experiment labels some contraction p with a multiset of cardinality $\sum_j m_j \cdot k^j$ (where $0 \leq m_j < k$ for any j), the integer j_0 is not related to the number of auxiliary doors in series anymore; it corresponds, in the case $m_{j_0} > 0$, with the existence of a box, whose some occurrence takes k^{j_0} copies of its content, having, among all its auxiliary doors, exactly m_{j_0} auxiliary doors that are, each of them, *the first one* (i.e. the deepest one) of a series of auxiliary doors immediately above the contraction p.

▶ **Example 26** (Continuation of Example 22)**.** We thus have $!_{f,1}[\mathcal{N}_1(f)] = \{o_2, o_4\}$; and indeed o_2 and o_4 are the boxes of depth 1 at depth 0 of $\mathcal{T}(f)[2] \equiv R$ (see Figure 1). Moreover we have $\mathcal{K}_{10,1}(\mathcal{T}(f)[1]) = \{p_1, p_4, p_5, p_6, p_7, o_2\}$ and $\mathcal{K}_{10,2}(\mathcal{T}(f)[1]) = \{p_4, p_5, p_6, p_7, o_4\}$; and indeed, in Figure 1, we have $\mathsf{im}(b_{\mathcal{T}(f)[2]}(o_2)) = \{p_1, p_4, p_5, p_6, p_7, o_2\}$ and $\mathsf{im}(b_{\mathcal{T}(f)[2]}(o_4)) = \{p_4, p_5, p_6, p_7, o_4\}$.

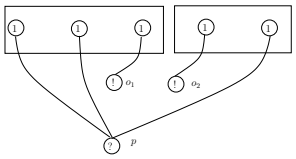


Figure 8 R_1.

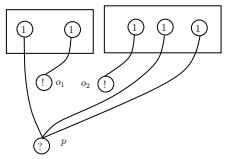


Figure 9 R_2.

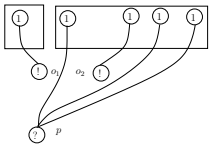


Figure 10 R_3.

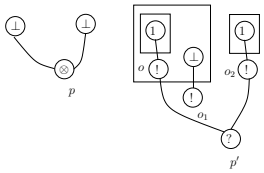


Figure 11 PS S.

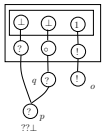


Figure 12 PS R'.

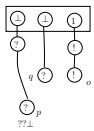


Figure 13 PS U.

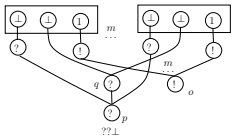


Figure 14 PS $\mathcal{T}(e')[1]$.

Figure 15 PS T.

As the following example shows, the information we obtain is already strong, but not strong enough.

▶ **Example 27.** The PS's R_1, R_2 and R_3 of Figures 8, 9 and 10 respectively have the same LPS. But if we know that $p \in \mathsf{im}(b_R(o_1))$, then we know that $R \neq R_3$. Still we are not able to distinguish between R_1 and R_2.

4.2 Connected components

In order to rebuild the content of the boxes, we introduce our notion of *connected component* (Definition 32), which uses the auxiliary notions of *substructure* (Definition 28) and *connected substructure* (Definition 31). A differential $\circ$-PS R is a substructure of a differential $\circ$-PS S (we write $R \sqsubseteq S$) if R is obtained from S by erasing some ports and wires. More precisely:

▶ **Definition 28.** Let R and S be two differential $\circ$-PS's. Let $\mathcal{Q} \subseteq \mathcal{P}_0^e(S)$. We write $R \sqsubseteq_{\mathcal{Q}} S$ to denote that $\mathcal{P}_0(R) \subseteq \mathcal{P}_0(S)$, $\mathcal{W}_0(R) = \{w \in \mathcal{W}_0(S) \cap (\mathcal{P}_0(R) \setminus \mathcal{Q}); t_{\mathcal{G}(S)}(w) \in \mathcal{P}_0(R)\}$, $l_{\mathcal{G}(R)} = l_{\mathcal{G}(S)}|_{\mathcal{P}_0(R)}$, $t_{\mathcal{G}(R)} = t_{\mathcal{G}(S)}|_{\mathcal{W}_0(R)}$, $\mathcal{L}(\mathcal{G}(R)) = \mathcal{L}(\mathcal{G}(S)) \cap \{w \in \mathcal{W}_0(S); t_{\mathcal{G}(S)}(w) \in \mathcal{P}_0^m(R)\}$, $\mathsf{T}_{\mathcal{G}(R)} = \mathsf{T}_{\mathcal{G}(S)}|_{\mathcal{P}_0(R)}$, $\mathcal{B}_0(R) = \mathcal{B}_0(S) \cap \mathcal{P}_0(R)$, $B_R = B_S|_{\mathcal{B}_0(R)}$ and $b_R = b_S|_{\mathcal{B}_0(R)}$. We write $R \sqsubseteq S$ if there exists $\mathcal{Q}$ such that $R \sqsubseteq_{\mathcal{Q}} S$.

▶ Remark 1. Let $R \sqsubseteq S$ and $\mathcal{Q} \subseteq \mathcal{P}_0^e(S)$. We have $R \sqsubseteq_{\mathcal{Q}} S$ iff $\mathcal{Q} \cap \mathcal{P}_0(R) \subseteq \mathcal{P}^{\mathsf{f}}(R)$.

▶ Remark 2. If $R, R' \sqsubseteq_{\mathcal{Q}} S$ and $\mathcal{P}_0(R) = \mathcal{P}_0(R')$, then $R = R'$.

Let us explain with the following example why we sometimes need to erase some wires (so the notion of $\sqsubseteq_\emptyset$ is not enough).

▶ **Example 29.** We need to erase some wires whenever there exist a box o and $p, q \in \mathsf{im}(b_{R'}(o))$ such that $t_{\mathcal{G}(R')}(q) = p$. Consider, for instance, Figure 12. If e' is a k-heterogeneous pseudo-experiment of R', then we want to be able to consider the PS U of Figure 13 as a substructure of $\mathcal{T}(e')[1]$, so we need to erase the wire q; we thus have $U \sqsubseteq_{\{p,q,o\}} \mathcal{T}(e')[1]$.

The relation $\frown_S$ formalizes the notion of "connectedness" between two ports of S at depth 0. But be aware that, here, "connected" has nothing to do with "connected" in the sense of [9]: here, any two doors of the same box are always "connected".

▶ **Definition 30.** Let S be a differential $\circ$-PS. We define the binary relation $\frown_S$ on $\mathcal{P}_0(S)$ as follows: for any $p, p' \in \mathcal{P}_0(S)$, we have $p \frown_S p'$ iff ($p \in \mathcal{W}_0(S)$ and $p' = t_{\mathcal{G}(S)}(p)$) or ($p' \in \mathcal{W}_0(S)$ and $p = t_{\mathcal{G}(S)}(p')$) or $((\exists o \in \mathcal{B}_0(S))\{p, p'\} \subseteq \mathsf{im}(b_S(o)))$.

▶ **Definition 31.** Let S and T be two differential $\circ$-PS's. Let $\mathcal{Q} \subseteq \mathcal{P}_0^e(S)$ such that $T \sqsubseteq_{\mathcal{Q}} S$. We write $T \trianglelefteq_{\mathcal{Q}} S$ if, for any $p, p' \in \mathcal{P}_0(T)$, there exists a finite sequence $(p_0, \ldots, p_n)$ of elements of $\mathcal{P}_0(T)$ such that $p_0 = p$, $p_n = p'$ and, for any $j \in \{0, \ldots, n-1\}$, we have $p_j \supset_S p_{j+1}$ and $(p_j \in \mathcal{Q} \Rightarrow j = 0)$.

▶ Remark 3. If $T \trianglelefteq_{\mathcal{Q}} S$, $\mathcal{Q}' \subseteq \mathcal{P}_0^e(S)$ and $\mathcal{P}_0(T) \cap \mathcal{Q}' \subseteq \mathcal{Q}$, then $T \trianglelefteq_{\mathcal{Q}'} S$.

The sets $\mathcal{S}_S^k((\mathcal{Q}, \mathcal{Q}_0))$ of "components T of S above $\mathcal{Q}$ and $\mathcal{Q}_0$ that are connected *via* other ports than $\mathcal{Q}$ and such that *cosize*$(T) < k$" will play a crucial role in the algorithm of the rebuilding of $\mathcal{T}(\overline{e})[i+1]$ from $\mathcal{T}(\overline{e})[i]$. The reader already knows that, here, "connected" has nothing to do with the "connected proof-nets" of [9]: there, the crucial tool used was rather the "bridges" that put together two doors of the same copy of some box only if they are connected in the LPS of the proof-net.

▶ **Definition 32.** Let $k \in \mathbb{N}$. Let S be a differential $\circ$-PS. Let $\mathcal{Q} \subseteq \mathcal{P}_0^e(S)$ and $\mathcal{Q}_0 \subseteq \mathcal{P}_0(S)$. We set $\mathcal{S}_S^k((\mathcal{Q}, \mathcal{Q}_0)) = \left\{ T \trianglelefteq_{\mathcal{Q}} S;\ \begin{array}{l} \textit{cosize}(T) < k \textit{ and } \mathcal{P}^{\mathsf{f}}(T) \subseteq \mathcal{Q} \cup \mathcal{Q}_0 \textit{ and} \\ \mathcal{P}_0(T) \setminus \mathcal{Q} \neq \emptyset \textit{ and } (\forall p \in \mathcal{P}_0(T))(\forall q \in \mathcal{P}_0(S)) \\ ((p \supset_S q \textit{ and } q \notin \mathcal{P}_0(T)) \Rightarrow p \in \mathcal{Q}) \end{array} \right\}$. We write also $\mathcal{S}_S^k(\mathcal{Q})$ instead of $\mathcal{S}_S^k((\mathcal{Q}, \emptyset))$ and $\mathcal{C}^k(S) = \mathcal{S}_S^k((\mathcal{P}_\circ^{\mathsf{f}}(S), \mathcal{P}_\bullet^{\mathsf{f}}(S)))$.

A port at depth 0 of S that is not in $\mathcal{Q}$ cannot belong to two different components:

▶ **Fact 33.** *Let $k \in \mathbb{N}$. Let S be a differential $\circ$-PS. Let $\mathcal{Q}, \mathcal{Q}_0 \subseteq \mathcal{P}_0(S)$. Let $T, T' \in \mathcal{S}_S^k((\mathcal{Q}, \mathcal{Q}_0))$ such that $(\mathcal{P}_0(T) \cap \mathcal{P}_0(T')) \setminus \mathcal{Q} \neq \emptyset$. Then $T = T'$.*

▶ **Example 34.** We have $\mathsf{Card}\left(\mathcal{S}_S^{10}(\{p_1, p_4, p_5, p_6, p_7, o_2\})\right) = 241$ and $\mathsf{Card}\left(\mathcal{S}_S^{10}(\{p_4, p_5, p_6, p_7, o_4\})\right) = 320$, where S is the PS of Figure 7.

The operator $\sum$ glues together several $\circ$-PS's that share only $\circ$-conclusions:

▶ **Definition 35.** Let $\mathcal{U}$ be a set of $\circ$-PS's. We say that $\mathcal{U}$ is *gluable* if, for any $R, S \in \mathcal{U}$ s.t. $R \neq S$, we have $\mathcal{P}_0(R) \cap \mathcal{P}_0(S) \subseteq \mathcal{P}_\circ^{\mathsf{f}}(R) \cap \mathcal{P}_\circ^{\mathsf{f}}(S)$. If $\mathcal{U}$ is gluable, then $\sum \mathcal{U}$ is the $\circ$-PS such that $\mathcal{P}_\circ^{\mathsf{f}}(\sum \mathcal{U}) = \bigcup \{\mathcal{P}_\circ^{\mathsf{f}}(R); R \in \mathcal{U}\}$ obtained by glueing all the elements of $\mathcal{U}$.

The set $\mathcal{C}^k(R)$ (for k big enough) is an alternative way to describe a $\circ$-PS R:

▶ **Fact 36.** *Let R be a $\circ$-PS. Let $k >$ cosize(R). We have $R = \sum \mathcal{C}^k(R)$.*

Definition 37 allows to formalize the operation of "putting a connected component inside a box", which will be useful for building the boxes of depth i of $\mathcal{T}(\overline{e})[i+1]$: from some *boxable* differential $\circ$-PS $R \sqsubseteq \mathcal{T}(\overline{e})[i]$, we build a $\circ$-PS $\overline{R}$ such that, for some $o \in \mathcal{B}_0^{=i}(\mathcal{T}(\overline{e})[i+1])$, there exists $T \in \mathcal{C}^k(B_{\mathcal{T}(\overline{e})[i+1]}(o))$ such that $T \simeq \overline{R}$.

▶ **Definition 37.** Let R be a differential $\circ$-PS. If $\mathcal{P}^{\mathsf{f}}(R) \subseteq \mathcal{P}_0^e(R)$, $\mathcal{P}^{\mathsf{f}}(R) \cap \mathcal{B}_0(R) = \emptyset$ and $\mathcal{P}_0^!(R) \setminus \mathcal{B}_0(R) \subseteq \mathcal{P}^{\mathsf{f}}(R)$, then one says that R is *boxable* and we define a $\circ$-PS $\overline{R}$ s.t. $\mathcal{P}_0(\overline{R}) \subseteq \mathcal{P}_0(R)$, $\mathcal{P}_\bullet^{\mathsf{f}}(\overline{R}) \subseteq \mathcal{W}_0(R)$ and $\mathcal{P}_\circ^{\mathsf{f}}(\overline{R}) = \mathcal{P}^{\mathsf{f}}(R) \cap \mathcal{P}_0(\overline{R})$ as follows:

- $\mathcal{P}_0(\overline{R}) = \mathcal{W}_0(R) \cup \bigcup_{o \in \mathcal{B}_0(R)} (\mathsf{im}(b_R(o)) \cap \mathcal{P}^{\mathsf{f}}(R))$;
- $\mathcal{W}_0(\overline{R}) = \{w \in \mathcal{W}_0(R); t_{\mathcal{G}(R)}(w) \in \mathcal{W}_0(R)\}$
- $l_{\mathcal{G}(\overline{R})}(p) = \left\{ \begin{array}{ll} l_{\mathcal{G}(R)}(p) & \textit{if } p \in \mathcal{W}_0(R); \\ \circ & \textit{otherwise;} \end{array} \right.$
- $t_{\mathcal{G}(\overline{R})} = t_{\mathcal{G}(R)}|_{\mathcal{W}_0(\overline{R})}$; $\mathcal{L}(\mathcal{G}(\overline{R})) = \mathcal{L}(\mathcal{G}(R)) \cap \mathcal{W}_0(\overline{R})$; $\mathcal{B}_0(\overline{R}) = \mathcal{B}_0(R)$; $b_{\overline{R}} = b_R$.

If $\mathcal{U}$ is a set of boxable differential $\circ$-PS's, then we set $\overline{\mathcal{U}} = \{\overline{R}; R \in \mathcal{U}\}$.

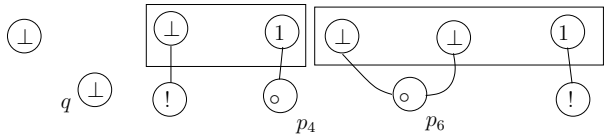


Figure 16 The ○-PS $\sum \overline{\mathcal{U}_{j_0}}$ of Example 39.

In the proof of the following proposition, we finally describe the complete algorithm leading from $\mathcal{T}(e)[i]$ to $\mathcal{T}(e)[i+1]$. Informally: for every $j_0 \in \mathcal{N}_i(e)$, for every equivalence class $\mathfrak{T} \in \mathcal{S}^k_{\mathcal{T}(e)[i]}(\mathcal{K}_{k,j_0}(\mathcal{T}(e)[i]))_{/\equiv}$, if $\mathsf{Card}(\mathfrak{T}) = \sum_{j\in\mathbb{N}} m_j \cdot k^j$ (with $0 \leq m_j < k$), then we remove $m_{j_0} \cdot k^{j_0}$ elements of $\mathfrak{T}$ from $\mathcal{G}(\mathcal{T}(e)[i])$ and we put m_{j_0} such elements inside the (new) box $!_{e,i}(j_0)$ of depth i. For every $j_0 \in \mathcal{N}_i(e)$, the set $\mathcal{U}_{j_0}$ of the proof is the union of the sets of such m_{j_0} elements for all the equivalence classes $\mathfrak{T} \in \mathcal{S}^k_{\mathcal{T}(e)[i]}(\mathcal{K}_{k,j_0}(\mathcal{T}(e)[i]))_{/\equiv}$.

▶ **Proposition 38.** *Let R and R' be two PS's. Let $k > cosize(R)$, $cosize(R')$, $\mathsf{Card}(\mathcal{B}(R))$, $\mathsf{Card}(\mathcal{B}(R'))$. Let e be a k-heterogeneous pseudo-experiment of R and let e' be a k-heterogeneous pseudo-experiment of R' s.t. $\mathcal{T}(e)[0] \equiv \mathcal{T}(e')[0]$. Then, for any $i \in \mathbb{N}$, we have $\mathcal{T}(e)[i] \equiv \mathcal{T}(e')[i]$.*

Proof (Sketch). By induction on i. We assume that $\mathcal{T}(e)[i] \equiv \mathcal{T}(e')[i]$. We set $\mathcal{M} = \mathcal{M}_i(e) = \mathcal{M}_i(e')$ and $\mathcal{N} = \mathcal{N}_i(e) = \mathcal{N}_i(e')$.

Let $S \equiv \mathcal{T}(e)[i]$. There is a bijection $! : \mathcal{M} \to \mathcal{P}^!_0(S) \setminus \mathcal{B}_0(S)$ such that, for any $j \in \mathcal{M}$, we have $(a_S \circ !)(j) = k^j$. For any $j \in \mathcal{N}$, we set $\mathcal{K}_j = \mathcal{K}_{k,j}(S)$ and $\mathcal{T}_j = \mathcal{S}^k_S(\mathcal{K}_j)$. We set $\mathcal{T} = \bigcup_{j\in\mathbb{N}} \mathcal{T}_j$. For any $T \in \mathcal{T}$, we define $(m^T_j)_{j\in\mathbb{N}} \in \{0,\dots,k-1\}^{\mathbb{N}}$ as follows: $\mathsf{Card}(\{T' \in \mathcal{T}; T \equiv T'\}) = \sum_{j\in\mathbb{N}} m^T_j \cdot k^j$. We set $\mathcal{P} = \{p \in \mathcal{P}_0(S);\ p \notin \bigcup_{j\in\mathcal{N}} \bigcup_{T \in \mathcal{S}^k_S(\mathcal{K}_j)} \mathcal{P}_0(T)\}$. For any $j \in \mathcal{N}$, we are given $\mathcal{U}_j \subseteq \mathcal{T}_j$ such that, for any $T \in \mathcal{T}_j$, we have $\mathsf{Card}(\{T' \in \mathcal{U}_j; T' \equiv T\}) = m^T_j$. Let S' be some differential PS such that $\mathcal{G}(S_{\restriction\mathcal{P}}) \sqsubseteq_\emptyset \mathcal{G}(S') \sqsubseteq_\emptyset \mathcal{G}(S)$, where $S_{\restriction\mathcal{P}}$ is the unique $S_0 \sqsubseteq_\emptyset S'$ s.t. $\mathcal{P}_0(S_0) = \mathcal{P}$, and:

- $\bigcup_{j_0\in\mathcal{N}} \mathcal{T}'_{j_0} \subseteq \bigcup_{j_0\in\mathcal{N}} \mathcal{T}_{j_0}$
- for any $T \in \mathcal{T}$, we have $\mathsf{Card}\left(\{T' \in \bigcup_{j\in\mathbb{N}} \mathcal{S}^k_{S'}(\mathcal{K}_j); T' \equiv T\}\right) = \sum_{j\notin\mathcal{N}} m^T_j \cdot k^j$;
- $\mathcal{B}_0(S') = (\mathcal{B}_0(S) \cap \mathcal{P}_0(S')) \cup\, ![\mathcal{N}]$
- for any $o_1 \in \mathcal{B}_0(S) \cap \mathcal{P}_0(S')$, we have $B_{S'}(o_1) = B_S(o_1)$ and $b_{S'}(o_1) = b_S(o_1)$
- for any $j \in \mathcal{N}$, there exists $\rho^j : B_{S'}(!(j)) \simeq \sum \overline{\mathcal{U}_j}$ such that $b_{S'}(!(j))(q) =$
$$\begin{cases} \mathcal{G}(\rho^j)(q) & \text{if } q \in \mathcal{P}^{\mathsf{f}}_\circ(B_{S'}(!(j)))\text{;} \\ t_{\mathcal{G}(S)}(\mathcal{G}(\rho^j)(q)) & \text{if } q \in \mathcal{P}^{\mathsf{f}}_\bullet(B_{S'}(!(j))). \end{cases}$$

Then one can show that $\mathcal{T}(e)[i+1] \equiv S' \equiv \mathcal{T}(e')[i+1]$. ◀

▶ **Example 39.** Consider Figure 7. We set $j_0 = 1$. We have $o_2 = !_{f,1}(j_0)$. We set $\mathcal{T}_{j_0} = \mathcal{S}^{10}_{\mathcal{T}(f)[1]}(\mathcal{K}_{10,1}(\mathcal{T}(f)[1]))$. Let T be the ○-PS of Figure 15. We have $T \in \mathcal{T}_{j_0}$ and $\mathsf{Card}(\{T' \in \mathcal{T}; T' \equiv T\}) = 1 \cdot 10^0 + 1 \cdot 10^1$. So we can take $\mathcal{U}_{j_0}$ in such a way that $\{T' \in \mathcal{U}_{j_0}; T' \equiv T\} = \{T\}$. There exists $\rho^{j_0} : B_{\mathcal{T}(f)[2]}(o_2) \simeq \sum \overline{\mathcal{U}_{j_0}}$ (see Figure 16) such that $b_{\mathcal{T}(f)[2]}(o_2)(q_2) = p_1 = t_{\mathcal{G}(\mathcal{T}(f)[1])}(\mathcal{G}(\rho^{j_0})(q_2))$ (see Figure 1): we set $\mathcal{G}(\rho^{j_0})(q_2) = q$.

4.3 Injectivity

▶ **Theorem 40.** *Let R and R' be two PS's s.t. $\mathcal{P}^{\mathsf{f}}(R) = \mathcal{P}^{\mathsf{f}}(R')$. If $[\![R]\!] = [\![R']\!]$, then $R \equiv R'$.*

Proof. We set $d = \max\{depth(R), depth(R')\}$. For any $k > 1$, there exist a k-heterogeneous experiment e of R and a k-heterogeneous experiment e' of R'[18] such that $e_{|\mathcal{P}^{\mathsf{f}}(R)} = e'_{|\mathcal{P}^{\mathsf{f}}(R')} \in [\![R]\!] \cap [\![R']\!]$. By Lemma 16, we have $\mathcal{T}(\overline{e})[0] \equiv \mathcal{T}(\overline{e'})[0]$. Therefore if $k > cosize(R), \mathsf{Card}(\mathcal{B}(R))$, then, by Proposition 38, we have $\mathcal{T}(\overline{e})[d] \equiv \mathcal{T}(\overline{e'})[d]$. Now, by Fact 17, we have $R = \mathcal{T}(\overline{e})[d] \equiv \mathcal{T}(\overline{e'})[d] = R'$. ◀

▶ Remark 4. From any 1-point of $[\![R]\!]$ (i.e. the result of some 1-experiment of R) one can recover $cosize(R)$ and $\mathsf{Card}(\mathcal{B}(R))$. This remark shows that for characterizing R, two points are enough: a 1-point of its interpretation, from which one can bound $cosize(R)$ and $\mathsf{Card}(\mathcal{B}(R))$, and a k-heterogeneous point of its interpretation with $k > cosize(R), \mathsf{Card}(\mathcal{B}(R))$.

▶ Remark 5. If we want to extend our theorem to PS's with axioms, then we assume that the interpretation of any ground type is an infinite set and we consider a k-heterogeneous experiment e that is "injective" in the sense that every atom (the atoms are the elements of the interpretations of the ground types) occurring in $e[\mathcal{P}^{\mathsf{f}}(R)]$ occurs exactly twice.

▶ Remark 6. In an untyped framework with axioms, we need to add the constraint on the injective k-heterogeneous point one considers to be $[\![R]\!]$-atomic, i.e. a point of $[\![R]\!]$ that cannot be obtained from another point of $[\![R]\!]$ by some substitution that is not a renaming. Atomic points are results of atomic experiments (experiments that label axioms with atoms) – the converse does not necessarily hold.

▶ Conclusion. We showed the injectivity of the relational semantics for MELL proof-nets by showing the injectivity of the Taylor expansion for cut-free MELL proof-nets, i.e. two different cut-free MELL proof-nets have different Taylor expansions; and, more precisely, we showed that, for any cut-free MELL proof-net, two simple differential nets in its Taylor expansion are enough to recover the entire proof-net. As a reviewer pointed out, it is worth noticing that the same proof should work in presence of cuts, i.e. our proof is a proof of a stronger result: the Taylor expansion of MELL proof-nets is injective (and two simple differential nets in the Taylor expansion of any MELL proof-net are enough to recover the entire proof-net).

References

1. Pierre Boudes. Unifying static and dynamic denotational semantics. Technical report, Institut de Mathématiques de Luminy, 2004.
2. Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In Josep Diaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science – 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2014.
3. Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation.* Ph.D. thesis, Université Paris 7, 1990.
4. Vincent Danos and Laurent Regnier. Proof-nets and the Hilbert space. In *Advances in Linear Logic*, volume 222 of *London Math. Soc. Lecture Note Ser.* Cambridge University Press, 1995.
5. René David and Walter Py. $\lambda\mu$-calculus and Böhm's theorem. *Journal of Symbolic Logic*, 66(1):407–413, March 2001.

[18] This is not necessary true for the multiset based coherent semantics.

6 Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. Ph.D. thesis, Université Aix-Marseille II, 2007.

7 Daniel de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. RR 6638, INRIA, 2008. To appear in Mathematical Structures in Computer Science.

8 Daniel de Carvalho, Michele Pagani, and Lorenzo Tortora de Falco. A semantic measure of the execution time in linear logic. *Theoretical Computer Science*, 412/20:1884–1902, 2011.

9 Daniel de Carvalho and Lorenzo Tortora de Falco. The relational model is injective for multiplicative exponential linear logic (without weakenings). *Annals of Pure and Applied Logic*, 163(9):1210–1236, 2012.

10 Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364:166–195, 2006.

11 Harvey Friedman. Equality between functionals. *Lecture Notes in Mathematics*, 453, 1975.

12 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

13 Giulio Guerrieri, Lorenzo Tortora De Falco, and Luc Pellissier. Injectivity of relational semantics for (connected) mell proof-nets via taylor expansion, 2014. URL: `https://hal.archives-ouvertes.fr/hal-00998847`.

14 Willem Heijltjes and Robin Houston. No proof nets for MLL with units: proof equivalence in MLL is pspace-complete. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 50:1–50:10. Association for Computing Machinery, 2014.

15 Dominic J. D. Hughes and Rob J. van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic (extended abstract). In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 1–10. IEEE Computer Society Press, 2003.

16 Gianfranco Mascari and Marco Pedicini. Head linear reduction and pure proof net extraction. *Theor. Comput. Sci.*, 135(1), 1994.

17 Satoshi Matsuoka. Weak typed böhm theorem on imll. *Annals of Pure and Applied Logic*, 145(1):37–90, 2007.

18 Damiano Mazza and Michele Pagani. The separation theorem for differential interaction nets. In *Proceedings of the 14th LPAR International Conference*, volume 4790 of *LNAI*. Springer, 2007.

19 Michele Pagani. Proofs, denotational semantics and observational equivalences in multiplicative linear logic. *Mathematical Structures in Computer Science*, 17(2):341–359, 2007. `doi:10.1017/S0960129506005652`.

20 Michele Pagani and Christine Tasson. The taylor expansion inverse problem in linear logic. In *Proceedings of the 24th ANNUAL IEEE Symposium on Logic in Computer Science*, 2009.

21 Simona Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:181–209, 1988.

22 Alexis Saurin. Separation with streams in the lambda-mu-calculus. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, 2005.

23 Lorenzo Tortora de Falco. *Réseaux, cohérence et expériences obsessionnelles*. Ph.D. thesis, Université Paris 7, January 2000.

24 Lorenzo Tortora de Falco. Obsessional experiments for linear logic proof-nets. *Mathematical Structures in Computer Science*, 13(6):799–855, 2003.

A With axioms (Remark 5)

With axioms, we need to slightly modify Definition 11, since different experiments can induce the same pseudo-experiment:

▶ **Definition 41.** Given an experiment e of some differential $\circ$-PS R, we define, by induction on $depth(R)$, a pseudo-experiment $\overline{e}$ of R as follows: $\overline{e}(\varepsilon) = (R, 1)$ and

$$\overline{e}(o) = \left\{ \overline{f}[\varepsilon \mapsto (B_R(o), i)]; f \in Supp(\mathcal{B}(e)(o)) \text{ and } 1 \leq i \leq \sum_{\substack{g \in Supp(\mathcal{B}(e)(o)) \\ \overline{g} = \overline{f}}} \mathcal{B}(e)(o)(g) \right\}$$

for any $o \in \mathcal{B}_0(R)$.

Notice that, if there is no axiom, Definitions 11 and **??** induce the same pseudo-experiment $\overline{e}$ for an experiment e.

B Untyped framework (Remark 6)

Since there is no type, we define (differential) ground-structures *via* the auxiliary definition of *(differential) pre-ground-structures*. We set $\mathfrak{T}' = \{\otimes, \parr, 1, \bot, !, ?, \circ, ax\}$.

▶ **Definition 42.** A *differential pre-ground-structure* is a 6-tuple $\mathcal{G} = (\mathcal{W}, \mathcal{P}, l, t, \mathcal{L}, \mathcal{A})$, where

- $\mathcal{P}$ is a finite set; the elements of $\mathcal{P}(\mathcal{G})$ are the *ports of* $\mathcal{G}$;
- l is a function $\mathcal{P} \to \mathfrak{T}'$;the element $l(p)$ of $\mathfrak{T}'$ is the *label of* p *in* $\mathcal{G}$;
- $\mathcal{W}$ is a subset of $\{p \in \mathcal{P}; l(p) \neq \circ\}$; the elements of $\mathcal{W}(\mathcal{G})$ are the *wires of* $\mathcal{G}$;
- t is a function $\mathcal{W} \to \{p \in \mathcal{P}; l(p) \notin \{1, \bot, ax\}\}$ such that, for any port p of $\mathcal{G}$, we have $(l(p) \in \{\otimes, \parr\} \Rightarrow \mathsf{Card}(\{w \in \mathcal{W}; t(w) = p\}) = 2)$; if $t(w) = p$, then w is a *premise of* p; the *arity* $a_{\mathcal{G}}(p)$ *of* p is the number of its premises;
- $\mathcal{L}$ is a subset of $\{w \in \mathcal{W}; l(t(w)) \in \{\otimes, \parr\}\}$ such that $(\forall p \in \mathcal{P})\ (l(p) \in \{\otimes, \parr\} \Rightarrow \mathsf{Card}(\{w \in \mathcal{L}; t(w) = p\}) = 1)$; if $w \in \mathcal{L}$ s.t. $t(w) = p$, then w is *the left premise of* p*;*
- and $\mathcal{A}$ is a partition of $\{p \in \mathcal{P}; l(p) = ax\}$ such that, for any $a \in \mathcal{A}$, $\mathsf{Card}(a) = 2$; the elements of $\mathcal{A}$ are the *axioms of* $\mathcal{G}$.

We set $\mathcal{W}(\mathcal{G}) = \mathcal{W}$, $\mathcal{P}(\mathcal{G}) = \mathcal{P}$, $l_{\mathcal{G}} = l$, $t_{\mathcal{G}} = t$, $\mathcal{L}(\mathcal{G}) = \mathcal{L}$, $\mathcal{A}(\mathcal{G}) = \mathcal{A}$. The set $\mathcal{P}^{\mathsf{f}}(\mathcal{G}) = \mathcal{P} \setminus \mathcal{W}$ is the set of *conclusions of* $\mathcal{G}$. For any $t \in \mathfrak{T}'$, we set $\mathcal{P}^t(\mathcal{G}) = \{p \in \mathcal{P}; l(p) = t\}$; we set $\mathcal{P}^m(\mathcal{G}) = \mathcal{P}^{\otimes}(\mathcal{G}) \cup \mathcal{P}^{\parr}(\mathcal{G})$; the set $\mathcal{P}^e(\mathcal{G})$ of *exponential ports of* $\mathcal{G}$ is $\mathcal{P}^!(\mathcal{G}) \cup \mathcal{P}^?(\mathcal{G}) \cup \mathcal{P}^{\circ}(\mathcal{G})$.

A *pre-ground-structure* is a differential pre-ground-structure $\mathcal{G}$ such that $\mathsf{im}(t_{\mathcal{G}}) \cap (\mathcal{P}^!(\mathcal{G}) \cup \mathcal{P}^{\circ}(\mathcal{G})) = \emptyset$.

A *differential ground-structure* (resp. a *ground-structure*) is a differential pre-ground structure (resp. a pre-ground structure) $\mathcal{G}$ such that the reflexive transitive closure $<_{\mathcal{G}}$ of the binary relation $<$ on $\mathcal{P}(\mathcal{G})$ defined by $p < p'$ *iff* $p = t_{\mathcal{G}}(p')$ is antisymmetric.

For the semantics of PS's, we are given a set A that does not contain any couple nor any 3-tuple and such that $* \notin A$. We define, by induction on n, the set $D_{A,n}$ for any $n \in \mathbb{N}$:

- $D_{A,0} = \{+, -\} \times (A \cup \{*\})$
- $D_{A,n+1} = D_{A,0} \cup (\{+, -\} \times D_{A,n} \times D_{A,n}) \cup (\{+, -\} \times \mathcal{M}_{fin}(D_{A,n}))$

We set $D_A = \bigcup_{n \in \mathbb{N}} D_{A,n}$.

Definition 44 is an adaptation of Definition 10 in an untyped framework.

▶ **Definition 43.** For any $\alpha \in D_A$, we define $\alpha^\perp \in D_A$ with $+^\perp = -$ and $-^\perp = +$:
- if $\alpha \in A \cup \{*\}$ and $\delta \in \{+,-\}$, then $(\delta,\alpha)^\perp = (\delta^\perp,\alpha)$;
- if $\alpha = (\delta,\alpha_1,\alpha_2)$ with $\delta \in \{+,-\}$ and $\alpha_1,\alpha_2 \in D_A$, then $\alpha^\perp = (\delta^\perp,{\alpha_1}^\perp,{\alpha_2}^\perp)$;
- if $\alpha = (\delta,[\alpha_1,\dots,\alpha_m])$ with $\delta \in \{+,-\}$ and $\alpha_1,\dots,\alpha_m \in D_A$, then $\alpha^\perp = (\delta^\perp,[{\alpha_1}^\perp,\dots,{\alpha_m}^\perp])$.

▶ **Definition 44.** For any differential $\circ$-PS R, we define, by induction on $\mathit{depth}(R)$ the set of *experiments of* R: it is is the set of triples $(R,e_{\mathcal{P}},e_{\mathcal{B}})$, where $e_{\mathcal{P}}$ is a function $\mathcal{P}_0(R) \to D_A \cup \mathcal{M}_{\mathit{fin}}(D_A)$ and $e_{\mathcal{B}}$ is a function which associates to every $o \in \mathcal{B}_0(R)$ a finite multiset of experiments of $B_R(o)$ such that
- for any $\{p,q\} \in \mathcal{A}(\mathcal{G}(R))$, we have $e_{\mathcal{P}}(p) = \alpha$, $e_{\mathcal{P}}(q) = \alpha^\perp$ for some $\alpha \in D_A$;
- for any $p \in \mathcal{P}_0^{\otimes}(R)$ (resp. $p \in \mathcal{P}_0^{\parr}(R)$), for any $w_1,w_2 \in \mathcal{W}_0(R)$ such that $t_{\mathcal{G}(R)}(w_1) = p = t_{\mathcal{G}(R)}(w_2)$, $w_1 \in \mathcal{L}(\mathcal{G}(R))$ and $w_2 \notin \mathcal{L}(\mathcal{G}(R))$, we have $e_{\mathcal{P}}(p) = (+,e_{\mathcal{P}}(w_1),e_{\mathcal{P}}(w_2))$ (resp. $e_{\mathcal{P}}(p) = (-,e_{\mathcal{P}}(w_1),e_{\mathcal{P}}(w_2))$);
- for any $p \in \mathcal{P}_0^1(R)$ (resp. $p \in \mathcal{P}_0^\perp(R)$), we have $e_{\mathcal{P}}(p) = (+,*)$ (resp. $e_{\mathcal{P}}(p) = (-,*)$);
- for any $p \in \mathcal{P}_0^e(R)$, we have $e(p) = \begin{cases} a & \text{if } p \in \mathcal{P}_\circ^{\mathsf{f}}(R); \\ (-,a) & \text{if } p \in \mathcal{P}_0^?(R); \\ (+,a) & \text{if } p \in \mathcal{P}_0^!(R); \end{cases}$ where

$$a = \sum_{\substack{w \in \mathcal{W}_0(R) \\ t_{\mathcal{G}(R)}(w) = p}} [e_{\mathcal{P}}(w)] + \sum_{o \in \mathcal{B}_0(R)} \sum_{e' \in \mathit{Supp}(e_{\mathcal{B}}(o))} \left(\sum_{\substack{q \in \mathcal{P}_\bullet^{\mathsf{f}}(B_R(o)) \\ b_R(o)(q) = p}} e_{\mathcal{B}}(o)(e') \cdot [{e'}_{\mathcal{P}}(q)] + \sum_{\substack{q \in \mathcal{P}_\circ^{\mathsf{f}}(B_R(o)) \\ b_R(o)(q) = p}} e_{\mathcal{B}}(o)(e') \cdot {e'}_{\mathcal{P}}(q) \right)$$

For any experiment $e = (R,e_{\mathcal{P}},e_{\mathcal{B}})$, we set $\mathcal{P}(e) = e_{\mathcal{P}}$ and $\mathcal{B}(e) = e_{\mathcal{B}}$.

For any differential $\circ$-PS R, we set $[\![R]\!]_A = \{\mathcal{P}(e)|_{\mathcal{P}^{\mathsf{f}}(R)}; e \text{ is an experiment of } R\}$.

▶ **Definition 45.** Let $r \in \mathcal{M}_{\mathit{fin}}(D_A)$. We say that r is *injective* if, for every $\gamma \in A$, there are at most two occurrences of γ in r.

For any set $\mathcal{P}$, for any function $x : \mathcal{P} \to D_A$, we say that x is *injective* if $\sum_{p \in \mathcal{P}} [x(p)]$ is injective. An experiment e of a differential $\circ$-PS S is said to be *injective* if $\mathcal{P}(e)|_{\mathcal{P}^{\mathsf{f}}(R)}$ is injective.

▶ **Definition 46.** Let $\sigma : A \to D_A$. For any $\alpha \in D_A$, we define $\sigma \cdot \alpha \in D_A$ as follows:
- if $\alpha \in A \cup \{*\}$, then $\sigma \cdot (+,\alpha) = \sigma(\alpha)$ and $\sigma \cdot (-,\alpha) = \sigma(\alpha)^\perp$;
- if $\delta \in \{+,-\}$ and $\alpha_1,\alpha_2 \in D_A$, then $\sigma \cdot (\delta,\alpha_1,\alpha_2) = (\delta,\sigma \cdot \alpha_1,\sigma \cdot \alpha_2)$;
- if $\delta \in \{+,-\}$ and $\alpha_1,\dots,\alpha_m \in D_A$, then $\sigma \cdot (\delta,[\alpha_1,\dots,\alpha_m]) = (\delta,[\sigma \cdot \alpha_1,\dots,\sigma \cdot \alpha_m])$.

For any set $\mathcal{P}$, for any function $x : \mathcal{P} \to D_A$, we define a function $\sigma \cdot x : \mathcal{P} \to D_A$ by setting: $(\sigma \cdot x)(p) = \sigma \cdot x(p)$ for any $p \in \mathcal{P}$.

▶ Remark 7. For any functions $\sigma,\sigma' : A \to D_A$, for any function $x : \mathcal{P} \to D_A$, we have $\sigma \cdot (\sigma' \cdot x) = (\sigma \cdot \sigma') \cdot x$.

▶ **Definition 47.** Let S be a differential $\circ$-PS. Let e be an experiment of S. Let $\sigma : A \to D_A$. We define, by induction of $\mathit{depth}(S)$, an experiment $\sigma \cdot e$ of S by setting $\mathcal{P}(\sigma \cdot e) = \sigma \cdot \mathcal{P}(e)$ and $\mathcal{B}(\sigma \cdot e)(o) = \sum_{e_1 \in \mathit{Supp}(\mathcal{B}(e)(o_1))} \mathcal{B}(e)(o_1)(e_1) \cdot [\sigma \cdot e_1]$ for any $o_1 \in \mathcal{B}_0(S)$.

Since we deal with untyped proof-nets, we cannot assume that the proof-nets are η-expanded and that experiments label the axioms only by atoms. That is why we introduce the notion of *atomic experiment*:

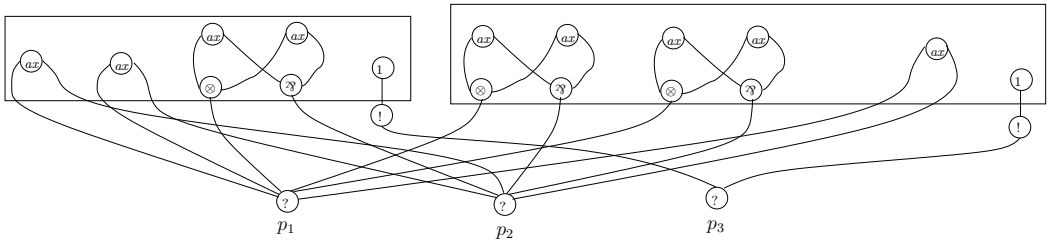


Figure 17 PS R''.

▶ **Definition 48.** For any differential ∘-PS R, we define, by induction on $depth(R)$, the set of *atomic experiments of* R: it is the set of experiments e of R such that

- for any $\{p,q\} \in \mathcal{A}(\mathcal{G}(R))$, we have $\mathcal{P}(e)(p), \mathcal{P}(e)(q) \in \{+,-\} \times A$;
- and, for any $o_1 \in \mathcal{B}_0(R)$, the multiset $\mathcal{B}(e)(o_1)$ is a multiset of atomic experiments of $B_R(o_1)$.

▶ **Definition 49.** Let $\mathcal{P}$ be a set. Let $\mathcal{D} \subseteq (D_A)^{\mathcal{P}}$. Let $x \in \mathcal{D}$, we say that x is $\mathcal{D}$-*atomic* if we have $(\forall \sigma \in (D_A)^A)(\forall y \in \mathcal{D})(\sigma \cdot y = x \Rightarrow (\forall \gamma \in At(y))\sigma(\gamma) \in A)$, where $At(y)$ is the set of atoms occurring in $\mathsf{im}(y)$.

We denote by $\mathcal{D}_{At}$ the subset of $\mathcal{D}$ consisting of the $\mathcal{D}$-atomic elements of $\mathcal{D}$.

For any PS R, any $[\![R]\!]_A$-atomic injective point is the result of some atomic experiment of R:

▶ **Fact 50.** *Let R be a ∘-PS. Let $x \in ([\![R]\!]_A)_{At}$ injective. Then there exists an atomic experiment e of R such that $e_{|\mathcal{P}^f(R)} = x$.*

The converse does not necessarily hold: for some PS's R, there are results of atomic injective experiments of R that are not $[\![R]\!]_A$-atomic. Indeed, consider Figure 17.

There exists an atomic injective experiment e of R'' such that

- $\mathcal{P}(e)(p_1) = (-, [(+,\gamma_1), \ldots, (+,\gamma_7), (+,(+,\gamma_8),(+,\gamma_9)), \ldots, (+,(+,\gamma_{22}),(+,\gamma_{23}))])$,
- $\mathcal{P}(e)(p_2) = (-, [(-,\gamma_1), \ldots, (-,\gamma_7), (-,(-,\gamma_8),(-,\gamma_9)), \ldots, (-,(-,\gamma_{22}),(-,\gamma_{23}))])$
- and $\mathcal{P}(e)(p_3) = (-, [(+, [(+,*),(+,*)])$, $(+, [(+,*),(+,*),(+,*)])])$,

where $\{\gamma_1, \ldots, \gamma_{23}\} \subseteq A$. But $e_{|\{p_1,p_2,p_3\}}$ is not in $([\![R'']\!]_A)_{At}$: there exists an atomic injective experiment e' of R' such that

- $\mathcal{P}(e')(p_1) = (-, [(+,\gamma_1), \ldots, (+,\gamma_8), (+,(+,\gamma_{10}),(+,\gamma_{11})), \ldots, (+,(+,\gamma_{22}),(+,\gamma_{23}))])$,
- $\mathcal{P}(e')(p_2) = (-, [(-,\gamma_1), \ldots, (-,\gamma_8), (-,(-,\gamma_{10}),(-,\gamma_{11})), \ldots, (-,(-,\gamma_{22}),(-,\gamma_{23}))])$
- and $\mathcal{P}(e')(p_3) = (-, [(+, [(+,*),(+,*)])$, $(+, [(+,*),(+,*),(+,*)])])$;

we set $\sigma(\gamma) = \begin{cases} \gamma & \text{if } \gamma \in A \setminus \{\gamma_8\}; \\ (+,(+,\gamma_8),(+,\gamma_9)) & \text{if } \gamma = \gamma_8; \end{cases}$ we have $\sigma \cdot e'_{|\{p_1,p_2,p_3\}} = e_{|\{p_1,p_2,p_3\}}$.

But it does not matter, because there are many enough atomic points:

▶ **Fact 51.** *Let R be a ∘-PS. For any $y \in [\![R]\!]_A$, there exist $x \in ([\![R]\!]_A)_{At}$ and $\sigma : A \to D_A$ such that $\sigma \cdot x = y$.*

Infinitary Proof Theory: the Multiplicative Additive Case

David Baelde[1], Amina Doumane[2], and Alexis Saurin[3]

1 **LSV, CNRS & ENS Cachan, Université Paris-Saclay, France**
`david.baelde@lsv.fr`
2 **IRIF, PPS, CNRS, Université Paris Diderot & Inria, France**
`amina.doumane@pps.univ-paris-diderot.fr`
3 **IRIF, PPS, CNRS, Université Paris Diderot & Inria, France**
`alexis.saurin@pps.univ-paris-diderot.fr`

Abstract

Infinitary and regular proofs are commonly used in fixed point logics. Being natural intermediate devices between semantics and traditional finitary proof systems, they are commonly found in completeness arguments, automated deduction, verification, etc. However, their proof theory is surprisingly underdeveloped. In particular, very little is known about the computational behavior of such proofs through cut elimination. Taking such aspects into account has unlocked rich developments at the intersection of proof theory and programming language theory. One would hope that extending this to infinitary calculi would lead, *e.g.*, to a better understanding of recursion and corecursion in programming languages. Structural proof theory is notably based on two fundamental properties of a proof system: cut elimination and focalization. The first one is only known to hold for restricted (purely additive) infinitary calculi, thanks to the work of Santocanale and Fortier; the second one has never been studied in infinitary systems. In this paper, we consider the infinitary proof system $\mu\mathsf{MALL}^\infty$ for multiplicative and additive linear logic extended with least and greatest fixed points, and prove these two key results. We thus establish $\mu\mathsf{MALL}^\infty$ as a satisfying computational proof system in itself, rather than just an intermediate device in the study of finitary proof systems.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Infinitary proofs, linear logic

Digital Object Identifier 10.4230/LIPIcs.CSL.2016.42

1 Introduction

Proof systems based on non-well-founded derivation trees arise naturally in logic, even more so in logics featuring fixed points. A prominent example is the long line of work on tableaux systems for modal μ-calculi, *e.g.*, [16, 24, 14, 11], which have served as the basis for analysing the complexity of the satisfiability problem, as well as devising practical algorithms for solving it. One key observation in such a setting, and many others, is that one needs not consider arbitrary infinite derivations but can restrict to *regular* derivation trees (also known as *circular* proofs) which are finitely representable and amenable to algorithmic manipulation. Because infinitary systems are easier to work with than the finitary proof systems (or axiomatizations) based on Kozen-Park (co)induction schemes, they are often found in completeness arguments for such finitary systems [16, 26, 27, 28, 15, 12]. We should note, however, that those arguments are far from being limited to translations from (regular) infinitary to finitary proofs, since such translations are very complex and only known to work in limited cases.

© David Baelde, Amina Doumanem and Alexis Saurin;
licensed under Creative Commons License CC-BY

There are many other uses of infinite (or regular) derivations, *e.g.*, to study the relationship between induction and infinite descent in first-order arithmetic [9], to generate invariants for program verification in separation logic [8], or as an intermediate between ludics' designs and proofs in linear logic with fixed points [5]. Last but not least, Santocanale introduced circular proofs [22] as a system for representing morphisms in μ-bicomplete categories [21, 23], corresponding to simple computations on (co)inductive data.

Surprisingly, despite the elegance and usefulness of infinitary proof systems, few proof theoretical studies are directly targeting these objects. More precisely, we are concerned with an analysis of proofs that takes into account their computational behaviour in terms of cut elimination. In other words, we would hope that the Curry-Howard correspondence extends nicely to infinitary proofs. In this line of proof-theoretical study, two main properties stand out: cut elimination and focalization; we shall see that they have been barely addressed in infinitary proof systems. The idea of cut elimination is as old as sequent calculus, and at the heart of the proof-as-program viewpoint, where the process of eliminating cuts in proofs is seen as computation. Considering logics with least and greatest fixed points, the computational behavior of induction and coinduction is recursion and corecursion respectively, two important and complex programming principles that would a logical understanding. Note that the many completeness results for infinitary proof systems (*e.g.*, for modal μ-calculi) only imply cut admissibility, but say nothing about the computational process of cut elimination. To our knowledge, leaving aside an early and very restrictive result of Santocanale [22], cut elimination has only been studied by Fortier and Santocanale [13] who considered an infinitary sequent calculus for lattice logic (purely additive linear logic with least and greatest fixed points) and showed that certain cut reductions converge to a limit cut-free derivation. Their proof involves a mix of combinatorial and topological arguments. So far, it has resisted attempts to extend it beyond the purely additive case. The second key property, much more recently identified than cut elimination, is focalization. It has appeared in the work of [3] on proof search and logic programming in linear logic, and is now recognized as one of the deep outcomes of linear logic, putting to the foreground the role of *polarity* in logic. In a way, focalization generalizes the reversibility results that are notably behind most deductive systems for classical μ-calculi, by bringing some key observations about non-reversible connectives. Besides its deep impact on proof search and logical frameworks, focalization resulted in important advances in all aspects of computational proof theory: in the game-semantical analysis of logic [17, 19], the understanding of evaluation order of programming languages, CPS translations, or semantics of pattern matching [10, 29], the space compression in computational complexity [25, 7], etc. Briefly, one can say that while proof nets have led to a better understanding of phenomena related to parallelism with proof-theoretical methods, polarities and focalization have led to a fine-grained understanding of sequentiality in proofs and programs. To the best of our knowledge, while reversibility has since long been a key-ingredient in completeness arguments based on infinitary proof systems, focalization has simply never been studied in such settings.

Organization and contributions of the paper. In this paper, we consider the logic $\mu\mathsf{MALL}$, that is multiplicative additive linear logic extended with least and greatest fixed point operators. It has been studied in finitary sequent calculus [4]: it notably enjoys free-cut elimination, and focalization has been shown to extend nicely (though not obviously) to it. We give in Section 2 a natural infinitary proof system for $\mu\mathsf{MALL}$, called $\mu\mathsf{MALL}^\infty$, which notably extends that of Santocanale and Fortier [13]. The system $\mu\mathsf{MALL}^\infty$ is also related to $\mu\mathsf{MALL}$ in the sense that any $\mu\mathsf{MALL}$ derivation can be turned into a $\mu\mathsf{MALL}^\infty$ proof, with

cuts. We study the focalization of $\mu\mathsf{MALL}^\infty$ in Section 3. We find out that, even though fixed point polarities are not forced in the finitary sequent calculus for $\mu\mathsf{MALL}$, they are uniquely determined in $\mu\mathsf{MALL}^\infty$. Despite some novel aspects due to the infinitary nature of our calculus, we are able to re-use the generic *focalization graph* argument [20] to prove that focalized proofs are complete. We then turn to cut elimination in Section 4 and show that (fair) cut reductions converge to an infinitary cut free derivation. We could not apply any standard cut elimination technique (*e.g.*, induction on formulas and proofs, reducibility arguments, topological arguments as in [13]) and propose instead an unusual argument in which a coarse truth semantics is used to show that the cut elimination process cannot go wrong. We also note here that, even for the regular fragment of $\mu\mathsf{MALL}^\infty$, it would be highly non-trivial to obtain cut elimination from the result for $\mu\mathsf{MALL}$, since it is not known whether regular $\mu\mathsf{MALL}^\infty$ derivations can be translated to $\mu\mathsf{MALL}$ derivations (even without requiring that this translation preserves the computational behaviour of proofs). We conclude in Section 5 with directions for future work. Technical details, proofs, and additional background material can be found in the long version of this paper [6].

2 $\mu\mathsf{MALL}$ and its infinitary proof system $\mu\mathsf{MALL}^\infty$

In this section we introduce multiplicative additive linear logic extended with least and greatest fixed point operators, and an infinitary proof system for it.

▶ **Definition 1.** Given an infinite set of propositional variables $\mathcal{V} = \{X, Y, \dots\}$, $\mu\mathsf{MALL}^\infty$ *pre-formulas* are built over the following syntax:

$$\varphi, \psi ::= \mathbf{0} \mid \top \mid \varphi \oplus \psi \mid \varphi \& \psi \mid \bot \mid \mathbf{1} \mid \varphi \parr \psi \mid \varphi \otimes \psi \mid \mu X.\varphi \mid \nu X.\varphi \mid X \quad \text{with} \quad X \in \mathcal{V}.$$

The connectives μ and ν bind the variable X in φ. From there, bound variables, free variables and capture-avoiding substitution are defined in a standard way. The subformula ordering is denoted $\leq$ and $\mathrm{fv}(\bullet)$ denotes free variables. Closed pre-formulas are simply called *formulas*. Note that negation is not part of the syntax, so that we do not need any positivity condition on fixed point expressions.

▶ **Definition 2.** *Negation* is the involution on pre-formulas written $\varphi^\perp$ and satisfying $(\varphi \parr \psi)^\perp = \psi^\perp \otimes \varphi^\perp$, $(\varphi \oplus \psi)^\perp = \psi^\perp \& \varphi^\perp$, $\bot^\perp = \mathbf{1}$, $\mathbf{0}^\perp = \top$, $(\nu X.\varphi)^\perp = \mu X.\varphi^\perp$, $X^\perp = X$.

Having $X^\perp = X$ might be surprising, but it is harmless since our proof system will only deal with closed pre-formulas. Our definition yields, *e.g.*, $(\mu X.X)^\perp = (\nu X.X)$ and $(\mu X.\mathbf{1} \oplus X)^\perp = (\nu X.X \& \bot)$, as expected [4]. Note that we also have $(\varphi[\psi/X])^\perp = \varphi^\perp[\psi^\perp/X]$.

Sequent calculi are sometimes presented with sequents as sets or multisets of formulas, but most proof theoretical observations actually hold in a stronger setting where one distinguishes between several *occurrences* of a formula in a sequent, which gives the ability to precisely *trace* the provenance of each occurrence. This more precise viewpoint is necessary, in particular, when one views proofs as programs. In this work, due to the nature of our proof system and because of the operations that we perform on proofs and formulas, it is also crucial to work with occurrences. There are several ways to formally treat occurrences; for the sake of clarity, we provide below a concrete presentation of that notion which is well suited for our needs.

▶ **Definition 3.** An *address* is a word over $\Sigma = \{l, r, i\}$, which stands for left, right and inside. We define a *duality* over Σ^* as the morphism satisfying $l^\perp = r$, $r^\perp = l$ and $i^\perp = i$. We say that α' is a *sub-address* of α when α is a prefix of α', written $\alpha \sqsubseteq \alpha'$. We say that α and β are *disjoint* when α and β have no upper bound wrt. $\sqsubseteq$.

$$\frac{\vdash F, \Gamma \quad \vdash G, \Gamma}{\vdash F \& G, \Gamma}\ (\&) \qquad \frac{\vdash F, G, \Gamma}{\vdash F \parr G, \Gamma}\ (\parr) \qquad \frac{\vdash F_i, \Gamma}{\vdash F_1 \oplus F_2, \Gamma}\ (\oplus_i) \qquad \frac{\vdash F, \Gamma \quad \vdash G, \Delta}{\vdash F \otimes G, \Gamma, \Delta}\ (\otimes)$$

$$\frac{}{\vdash \top, \Gamma}\ (\top) \qquad \frac{\vdash \Gamma}{\vdash \bot, \Gamma}\ (\bot) \qquad \text{(no rule for } \mathbf{0}) \qquad \frac{}{\vdash \mathbf{1}}\ (\mathbf{1})$$

$$\frac{\vdash F[\mu X.F/X], \Gamma}{\vdash \mu X.F, \Gamma}\ (\mu) \qquad \frac{\vdash G[\nu X.G/X], \Gamma}{\vdash \nu X.G, \Gamma}\ (\nu) \qquad \frac{F \equiv G}{\vdash F, G^\perp}\ (\mathsf{Ax}) \qquad \frac{\vdash \Gamma, F \quad \vdash F^\perp, \Delta}{\vdash \Gamma, \Delta}\ (\mathsf{Cut})$$

Figure 1 Rules of the proof system $\mu\mathsf{MALL}^\infty$.

▶ **Definition 4.** A *(pre)formula occurrence* (denoted by F, G, H) is given by a (pre)formula φ and an address α, and written φ_α. We say that occurrences are *disjoint* when their addresses are. The occurrences φ_α and ψ_β are *structurally equivalent*, written $\varphi_\alpha \equiv \psi_\beta$, if $\varphi = \psi$. Operations on formulas are extended to occurrences as follows: $(\varphi_\alpha)^\perp = (\varphi^\perp)_{\alpha^\perp}$; for any $\star \in \{\parr, \otimes, \oplus, \&\}$, $F \star G = (\varphi \star \psi)_\alpha$ if $F = \varphi_{\alpha l}$ and $G = \psi_{\alpha r}$; for any $\sigma \in \{\mu, \nu\}$, $\sigma X.F = (\sigma X.\varphi)_\alpha$ if $F = \varphi_{\alpha i}$; we also allow ourselves to write units as formula occurrences without specifying their address, which can be chosen arbitrarily. Finally, *substitution of occurrences* forgets addresses: $(\varphi_\alpha)[\psi_\beta/X] = (\varphi[\psi/X])_\alpha$.

▶ **Example 5.** Let $F = \varphi_{\alpha l}$ and $G = \psi_{\alpha r}$. We have, on the one hand, $(F \otimes G)^\perp = ((\varphi \otimes \psi)_\alpha)^\perp = (\psi^\perp \parr \varphi^\perp)_{\alpha^\perp}$ and, on the other hand, $G^\perp \parr F^\perp = (\psi^\perp)_{\alpha^\perp l} \parr (\varphi^\perp)_{\alpha^\perp r} = (\psi^\perp \parr \varphi^\perp)_{\alpha^\perp}$. Thus, $(F \otimes G)^\perp = G^\perp \parr F^\perp$. We could have designed our system to obtain $(F \otimes G)^\perp = F^\perp \parr G^\perp$ instead; this choice is inessential for the present work but makes our definitions suitable, in principle, for a treatment of non-commutative logic.

▶ **Definition 6.** The *Fischer-Ladner closure* of a formula occurrence F, denoted by $\mathrm{FL}(F)$, is the least set of formula occurrences such that $F \in \mathrm{FL}(F)$ and, whenever $G \in \mathrm{FL}(F)$,

- $G_1, G_2 \in \mathrm{FL}(F)$ if $G = G_1 \star G_2$ for any $\star \in \{\oplus, \&, \parr, \otimes\}$;
- $B[G/X] \in \mathrm{FL}(F)$ if $G = \sigma X.B$ for $\sigma \in \{\nu, \mu\}$.

We say that G is a *sub-occurrence* of F if $G \in \mathrm{FL}(F)$. Note that, for any F and α, there is at most one φ such that φ_α is a sub-occurrence of F.

We are now ready to introduce our infinitary sequent calculus. Details regarding formula occurrences can be ignored at first read, and will only make full sense when one starts permuting inferences and eliminating cuts.

▶ **Definition 7.** A *sequent*, written $\vdash \Gamma$, is a finite set of pairwise disjoint formula occurrences. A *pre-proof* of $\mu\mathsf{MALL}^\infty$ is a possibly infinite tree, coinductively generated by the rules of Figure 1, subject to the following conditions: any two formulas occurrences appearing in different branches must be disjoint except if the branches first differ right after a $(\&)$ inference; if φ_α and $\psi_{\alpha^\perp}$ occur in a pre-proof, they must be the respective sub-occurrences of the formula occurrences F and $F^\perp$ introduced by a (Cut) rule.

The disjointness condition on sequents ensures that two formula occurrences from the same sequent will never engender a common sub-occurrence, *i.e.*, we can define traces uniquely. The disjointness condition on pre-proofs is there to ensure that the proof transformations used in focusing and cut elimination preserve the disjointness condition on sequents. Note that these conditions are not restrictive. Clearly, the condition on sequents never prevents the (backwards) application of a propositional rule. Moreover, there is an infinite supply of disjoint addresses, *e.g.*, $\{\, r^n l \;:\; n > 0 \,\}$. One may thus pick addresses from that supply for

the conclusion sequent of the derivation, and then carry the remaining supply along proof branches, splitting it on branching rules, and consuming a new address for cut rules.

Pre-proofs are obviously unsound: the pre-proof schema shown below allows one to derive any formula. In order to obtain proper proofs from pre-proofs, we will add a validity condition. This condition will reflect the nature of our two fixed point connectives.

$$\dfrac{\dfrac{\vdots}{\vdash \mu X.X}\,(\mu) \qquad \dfrac{\vdots}{\vdash \nu X.X, F}\,(\nu)}{\vdash F}\,(\mathsf{Cut})$$

▶ **Definition 8.** Let $\gamma = (s_i)_{i\in\omega}$ be an infinite branch in a pre-proof of $\mu\mathsf{MALL}^\infty$. A *thread* t in γ is a sequence of formula occurrences $(F_i)_{i\in\omega}$ with $F_i \in s_i$ and $F_i \sqsubseteq F_{i+1}$. The set of formulas that occur infinitely often in $(F_i)_{i\in\omega}$ (when forgetting addresses) admits a minimum wrt. the subformula ordering, denoted by $\min(t)$. A thread t is *valid* if $\min(t)$ is a ν formula and the thread is not eventually constant, *i.e.*, the formulas F_i are always eventually principal.

▶ **Definition 9.** The *proofs* of $\mu\mathsf{MALL}^\infty$ are those pre-proofs in which every infinite branch contains a valid thread.

This validity condition has its roots in parity games and is very natural for infinitary proof systems with fixed points. It is somehow independent of the ambient logic, and only deals with fixed points. It is commonly found in deductive systems for modal μ-calculi: see [11] for a closely related presentation, which yields a sound and complete sequent calculus for linear time μ-calculus. The validity conditions of Santocanale's circular proofs [22, 13], with and without cut, are also instances of the above notion.

In the rest of the paper, we work mostly with formula occurrences and will often simply call them formulas when it is not ambiguous. As usual in sequent calculus, (Ax) on a formula F can be expanded into axioms on its immediate subformulas. Repeating this process, one obtains an axiom-free and valid proof of the original sequent. In fact, this construction yields a *regular* derivation tree, the simplest kind of finitely representable infinite derivation.

▶ **Proposition 10.** *Rule* (Ax) *is admissible in* $\mu\mathsf{MALL}^\infty$.

This basic observation, proved in [6], justifies that the (Ax) rule will be ignored in the rest of the paper. In particular, we consider that axioms are expanded away before dealing with cut elimination. Our system $\mu\mathsf{MALL}^\infty$ is naturally equipped with the cut elimination rules of MALL, extended with the obvious principal and auxiliary rules for fixed point connectives (we do not show symmetric cases):

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, F[\mu X.F/X]}{\vdash \Gamma, \mu X.F}\,(\mu) \qquad \dfrac{\vdash F^\perp[\nu X.F^\perp/X], \Delta}{\vdash \nu X.F^\perp, \Delta}\,(\nu)}{\vdash \Gamma, \Delta}\,(\mathsf{Cut})}{\downarrow}$$

$$\dfrac{\vdash \Gamma, F[\mu X.F/X] \qquad \vdash F^\perp[\nu X.F^\perp/X], \Delta}{\vdash \Gamma, \Delta}\,(\mathsf{Cut})$$

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, F[\mu X.F/X], G}{\vdash \Gamma, \mu X.F, G}\,(\mu) \qquad \vdash G^\perp, \Delta}{\vdash \Gamma, \mu X.F, \Delta}\,(\mathsf{Cut})}{\downarrow}$$

$$\dfrac{\dfrac{\vdash \Gamma, F[\mu X.F/X], G \qquad \vdash G^\perp, \Delta}{\vdash \Gamma, F[\mu X.F/X], \Delta}\,(\mathsf{Cut})}{\vdash \Gamma, \mu X.F, \Delta}\,(\mu)$$

Natural numbers may be expressed as $\varphi_{\mathsf{nat}} := \mu X.\mathbf{1}\oplus X$. Occurrences of that formula will be denoted N, N', etc. We give below a few examples of proofs/computations on natural numbers, shown using two sided sequents for clarity: $F_1, \ldots, F_n \vdash \Gamma$ should be read as

$\vdash \Gamma, F_1^\perp, \ldots, F_n^\perp$ as usual. The proof π_{succ}, shown below, computes the successor on natural numbers: if we cut it against a (necessarily finite) cut-free proof of N we obtain after a finite number of cut elimination steps a proof of N' which is the right injection (rule (μ) followed by $(\oplus_2)$, which represents the successor) of the original proof of N, relocated at the address of N''.

$$\dfrac{\dfrac{\dfrac{}{N \vdash N''}\,(\mathsf{Ax})}{N \vdash \mathbf{1}\oplus N''}\,(\oplus_2)}{N \vdash N'}\,(\mu)$$

Consider now the following pre-proof, called π_{dup}:

$$\dfrac{\dfrac{\dfrac{}{\vdash N_1}\,(\mu),(\oplus_1),(\mathbf{1}) \qquad \dfrac{}{\vdash N_2}\,(\mu),(\oplus_1),(\mathbf{1})}{\mathbf{1} \vdash N_1 \otimes N_2}\,(\perp),(\otimes) \qquad \dfrac{(\star)}{N' \vdash N_1' \otimes N_2'} \qquad \dfrac{\pi_{\mathsf{succ}} \quad \pi_{\mathsf{succ}}}{N_1' \otimes N_2' \vdash N_1 \otimes N_2}\,(\parr),(\otimes)}{N' \vdash N_1 \otimes N_2}\,(\mathsf{Cut})}{(\star) \quad N \vdash N_1 \otimes N_2}\,(\nu),(\&)$$

Here, $(\star)$ represents the cyclic repetition of the same proof, on a structurally equivalent sequent (same formulas, new addresses). The resulting pre-proof has exactly one infinite branch, validated by the thread starting with N. If we cut that proof against an arbitrary cut-free proof of N, and perform cut elimination steps, we obtain in finite time a cut-free proof of $N_1 \otimes N_2$ which consists of two copies (up-to addresses) of the original proof of N.

Now let $\varphi_{\mathsf{stream}} = \nu X.\varphi_{\mathsf{nat}} \otimes X$ be the formula representing infinite streams of natural numbers, whose occurrences will be denoted by S, S', etc. Let us consider the derivation shown below, where F is an arbitrary, useless formula occurrence for illustrative purposes.

$$\dfrac{\dfrac{\pi_{\mathsf{dup}}}{N \vdash N_1 \otimes N_2} \qquad \dfrac{\dfrac{\dfrac{}{N_1 \vdash N'}\,(\mathsf{Ax}) \qquad \dfrac{\dfrac{\pi_{\mathsf{succ}}}{N_2 \vdash N''} \qquad \dfrac{(\star)}{N'', F \vdash S'}}{N_2, F \vdash S'}\,(\mathsf{Cut})}{N_1, N_2, F \vdash N' \otimes S'}}{N_1 \otimes N_2, F \vdash N' \otimes S'}}{N, F \vdash N' \otimes S'}\,(\mathsf{Cut})}{(\star) \quad N, F \vdash S}$$

It is a valid proof thanks to the thread on S. By cut elimination, the computational behaviour of that proof is to take a natural number n, and some irrelevant f, and compute the stream $n :: (n+1) :: (n+2) :: \ldots$. However, unlike in the two previous examples, the result of the computation is not obtained in finite time; instead, we are faced with a productive process which will produce any finite prefix of the stream when given enough time. The presence of the useless formula F illustrates here that weakening may be admissible in $\mu\mathsf{MALL}^\infty$ under some circumstances, and that cutting against some formulas (F in this case) will form a redex that will be delayed forever. These subtleties will show up in the next two sections, devoted to showing our two main results.

3 Focalization

Focalization in linear logic. MALL connectives can be split in two classes: *positive* $(\otimes, \oplus, \mathbf{0}, \mathbf{1})$ and *negative* $(\parr, \&, \top, \perp)$ connectives. The distinction can be easily understood in terms of proof search: negative inferences $(\parr), (\&), (\top)$ and $(\perp)$ are *reversible* (meaning

that provability of the conclusion transfers to the premisses) while positive inferences require choices (splitting the context in $(\otimes)$ or choosing between $(\oplus_1)$ and $(\oplus_2)$ rules) resulting in a possible of loss of provability. Still, positive inferences satisfy the *focalization* property [3]: in any provable sequent containing no negative formula, some formula can be chosen as a *focus*, hereditarily selecting its positive subformulas as principal formulas until a negative subformula is reached. It induces the following complete proof search strategy:

Sequent Γ *contains a negative* formula	Sequent Γ *contains no negative* formula
Choose any negative formula (*e.g.*, the leftmost one) and decompose it using the only possible negative rule.	Choose some positive formula and decompose it (and its subformulas) hereditarily until we get to atoms or negative subformulas.

Focalization graphs. Focused proofs are complete for proofs, not only provability: any linear proof is equivalent to a focused proof, up to cut-elimination. Indeed, focalization can be proved by means of proof transformations [18, 20, 7] preserving the denotation of the proof. A flexible, modular method for proving focalization that we shall apply in the next sections has been introduced by Miller and the third author [20] and relies on *focalization graphs*. The heart of the focalization graph proof technique relies on the fact the positive inference, while not reversible, all permute with each other. As a consequence, if the positive layer of some positive formula is completely decomposed within the lowest part of the proof, below any negative inference, then it can be taken as a focus. Focalization graphs ensure that it is always possible: their acyclicity provides a source which can be taken as a focus.

Focusing infinitary proofs. The infinitary nature of our proofs interferes with focalization in several ways. First, while in μMALL μ and ν can be set to have an arbitrary polarity, we will see that in $\mu\mathsf{MALL}^\infty$, ν must be negative. Second, permutation properties of the negative inferences, which can be treated locally in μMALL, now require a global treatment due to infinite branches. Last, focalization graphs strongly rely on the finiteness of maximal positive subtrees of a proof: this invariant must be preserved in $\mu\mathsf{MALL}^\infty$. For simplicity reasons, we restrict our attention to cut-free proofs in the rest of this section. The result holds for proofs with cuts thanks to the usual trick of viewing cuts as $\otimes$.

3.1 Polarity of connectives

Let us first consider the question of polarizing $\mu\mathsf{MALL}^\infty$ connectives. Unlike in μMALL, we are not free to set the polarity of fixed points formulas: consider the proof π of sequent $\vdash \mu X.X, \nu Y.Y$ which alternates inferences (ν) and (μ). Assigning opposite polarities to dual formulas (an invariant necessary to define properly cut-elimination in focused proof systems), this sequent contains a negative formula; each polarization of fixed points induces one focused pre-proof, either π_μ which always unrolls μ or π_ν which repeatedly unrolls ν. Only π_ν happens to be valid, leaving but one possible choice, $\nu X.F$ negative and $\mu X.F$ positive, resulting in the following polarization:

▶ **Definition 11.** *Negative formulas* are formulas of the form $\nu X.F$, $F \parr G$, $F \& G$, $\bot$ and $\top$, *positive formulas* are formulas of the form $\mu X.F$, $F \otimes G$, $F \oplus G$, $\mathbf{1}$ and $\mathbf{0}$. A $\mu\mathsf{MALL}^\infty$ sequent containing only positive formulas is said to be *positive*. Otherwise, it is *negative*.

The following proposition will be useful in the following:

▶ **Proposition 12.** *An infinite branch of a pre-proof containing only negative (resp. positive) rules is always valid (resp. invalid).*

3.2 Reversibility of negative inferences

The example shown below with $F = \nu X.(X \& X) \oplus \mathbf{0}$ shows that, unlike in (MA)LL, negative inferences cannot be permuted down locally: no occurrence of a negative inference ($\parr$) on $P \parr Q$ can be permuted below a ($\&$) since it is never available in the left premise.

$$\dfrac{\dfrac{\dfrac{\dfrac{(\star)}{\vdash F, P \parr Q} \qquad \dfrac{\dfrac{\pi'}{\vdash F, P, Q}}{\vdash F, P \parr Q}\,(\parr)}{\vdash F \& F, P \parr Q}\,(\&)}{\vdash (F \& F) \oplus \mathbf{0}, P \parr Q}\,(\oplus_1)}{(\star) \ \vdash F, P \parr Q}\,(\nu)$$

We shall thus introduce a global proof transformation (which could be realized by means of cut, as is usual). In order to define this transformation at once for all negative connectives, we rely on the uniform structure of negative inferences, which can be written as follows:

$$\dfrac{(\vdash \Gamma, \mathcal{N}_i^N)_{1 \leq i \leq n}}{\vdash \Gamma, N}\,(\mathsf{r_N})$$

Sub-occurrence families of N are then defined as $\mathcal{N}(N) = (\mathcal{N}_i^N)_{1 \leq i \leq n}$, its *slicing index* being $\mathsf{sl}(N) = \#\mathcal{N}(N)$.

N	$F_1 \parr F_2$	$\bot$	$F_1 \& F_2$	$\top$	$\nu X.F$
$\mathcal{N}(N)$	$\{1 \mapsto \{F_1, F_2\}\}$	$\{1 \mapsto \emptyset\}$	$\{1 \mapsto \{F_1\}, 2 \mapsto \{F_2\}\}$	$\emptyset$	$\{1 \mapsto \{F[\nu X.F/X]\}\}$

We can now define, in two steps, how to transform any proof π into a proof $\mathsf{rev}(\pi)$ where all negative inferences are reversed.

▶ **Definition 13** ($\pi(i, N)$)**.** Let π be a proof of $\vdash \Gamma$ of last rule (r) and premises $\pi_1, \ldots, \pi_n$. If $1 \leq i \leq \mathsf{sl}(N)$, we define $\pi(i, N)$ coinductively:

- if N does not occur in $\vdash \Gamma$, then $\pi(i, N) = \pi$;
- if r is the inference on N, then $\pi(i, N) = \pi_i$ (which is legal since in this case $n = \mathsf{sl}(N)$);
- if r is not the inference on N, then

$$\pi(i, N) = \dfrac{\pi_1(i, N) \quad \ldots \quad \pi_n(i, N)}{\vdash \Gamma, \mathcal{N}_i^N}\,(\mathsf{r}).$$

▶ **Definition 14** ($\mathsf{rev}(\pi)$)**.** Let π be a $\mu\mathsf{MALL}^\infty$ proof of $\vdash \Gamma$. Then $\mathsf{rev}(\pi)$ is a pre-proof non-deterministically defined as π if $\vdash \Gamma$ is positive and, otherwise, when $N \in \Gamma$ and $n = \mathsf{sl}(N)$, as

$$\mathsf{rev}(\pi) = \dfrac{\mathsf{rev}(\pi(1, N)) \quad \ldots \quad \mathsf{rev}(\pi(n, N))}{\vdash \Gamma}\,(\mathsf{r_N}).$$

Reversed proofs formalize the requirement for the whole negative layer to be reversed:

▶ **Definition 15.** *Reversed pre-proofs* are defined to be the largest set of pre-proofs such that: (i) every pre-proof of a positive sequent is reversed; (ii) a pre-proof of a negative sequent is reversed if it ends with a negative inference and if each of its premises is reversed.

▶ **Example 16.** We illustrate rev on the proof π starting this subsection. We have $\mathsf{sl}(P \parr Q) = 1$ and:

$$\mathsf{rev}(\pi) \quad = \quad \dfrac{\pi(1, P \parr Q)}{\vdash F, P \parr Q}\,(\parr) \quad = \quad \dfrac{\dfrac{\dfrac{(\star)}{\vdash F, P, Q} \qquad \dfrac{\pi'}{\vdash F, P, Q}\,(\&)}{\vdash F \& F, P, Q}}{\dfrac{\vdash (F \& F) \oplus \mathbf{0}, P, Q}{(\star) \;\vdash F, P, Q}\,(\nu)}\,(\oplus_1)}{\vdash F, P \parr Q}\,(\parr)$$

▶ **Theorem 17.** *If π is a μMALL$^\infty$ proof, then* rev(π) *is a reversed proof of the same sequent.*

3.3 Focalization Graph

In this section, we adapt the focalization graphs introduced in [20] to our setting. Considering the permutability properties of positive inferences in μMALL$^\infty$, finiteness of positive trunks and acyclicity of focalization graphs will be sufficient to make the proof technique of [20] applicable. In order to illustrate this subsection, an example is fully explained in [6].

▶ **Definition 18** (Positive trunk, positive border, active formulas)**.** Let π be a μMALL$^\infty$ proof of $\mathcal{S}$. The *positive trunk* π^+ of π is the tree obtained by cutting (finite or infinite) branches of π at the first occurrence of a negative rule. The *positive border* of π is the collection of lowest sequents in π which are conclusions of negative rules. *P-active* formulas of π are those formulas of $\mathcal{S}$ which are principal formulas of an inference in π^+.

▶ **Proposition 19.** *The positive trunk of a μMALL$^\infty$ proof is always finite.*

▶ **Definition 20** (Focalization graph)**.** Given a μMALL$^\infty$ proof π, we define its *focalization graph* $\mathcal{G}(\pi)$ to be the graph whose vertices are the P-active formulas of π and such that there is an edge from F to G iff there is a sequent $\mathcal{S}'$ in the positive border containing a negative sub-occurrence F' of F and a positive sub-occurrence G' of G.

μMALL$^\infty$ positive inferences are those of MALL extended with (μ) which is not branching: this ensures both that any two positive inferences permute and that the proof of acyclicity of MALL focalization graphs can easily be adapted, from which we conclude that:

▶ **Proposition 21.** *Focalization graphs are acyclic.*

Acyclicity of the focalization graph implies in particular that it has a source, that is a formula P of the conclusion sequent such that whenever one of its subformulas F appears in a border sequent, F is negative. This remark, together with the fact that the trunk is finite ensures that the positive layer of P is completely decomposed in the positive trunk.

▶ **Definition 22** (foc(π, P))**.** Let π be a μMALL$^\infty$ proof of $\vdash \Gamma, P$ with P a source of π's focalization graph. One defines foc(π, P) as the μMALL$^\infty$ proof obtained by permuting down all the positive inferences on P and its positive subformulas (all occurring in π^+).

▶ **Proposition 23.** *Let $\mathcal{S}$ be a lowest sequent of* foc(π, P) *which is not the conclusion of a rule on a positive subformula of P. Then $\mathcal{S}$ contains exactly one subformula of P, which is negative.*

3.4 Productivity and validity of the focalization process

Reversibility of the negative inferences and focalization of the positive inferences allows one to consider the following (non-deterministic) proof transformation process:

Focalization Process: Let π be a $\mu\mathsf{MALL}^\infty$ proof of $\mathcal{S}$. Define $\mathsf{Foc}(\pi)$ as follows:

- **Asynchronous phase:** If $\mathcal{S}$ is negative, transform π into $\mathsf{rev}(\pi)$ which is reversed. At least one negative inference has been brought to the root of the proof. Apply (corecursively) the synchronous phase to the proofs rooted in the lowest positive sequents of $\mathsf{rev}(\pi)$.
- **Synchronous phase:** If $\mathcal{S}$ is positive, let $P \in \mathcal{S}$ be a source of the associated focalization graph. Transform π into a proof $\mathsf{foc}(\pi, P)$. At least one positive inference on P has been brought to the root of the proof. Apply (corecursively) the asynchronous phase to the proofs rooted in the lowest negative sequents of $\mathsf{foc}(\pi, P)$.

Each of the above phases produces one non-empty phase, the above process is thus productive. It is actually a pre-proof thanks to Theorem 17 and by definition of $\mathsf{foc}(\pi, P)$. It remains to show that the resulting pre-proof is actually a proof. The following property is easily seen to be preserved by both transformations foc and rev and thus holds for $\mathsf{Foc}(\pi)$:

▶ **Proposition 24.** *Let π be a $\mu\mathsf{MALL}^\infty$ proof, r a positive rule occurring in π and r' be a negative rule occurring below r in π. If r occurs in $\mathsf{Foc}(\pi)$, then r' occurs in $\mathsf{Foc}(\pi)$, below r.*

▶ **Lemma 25.** *For any infinite branch γ of $\mathsf{Foc}(\pi)$ containing an infinite number of positive rules, there exists an infinite branch in π containing infinitely many positive rules of γ.*

▶ **Theorem 26.** *If π is a $\mu\mathsf{MALL}^\infty$ proof then $\mathsf{Foc}(\pi)$ is also a $\mu\mathsf{MALL}^\infty$ proof.*

Proof sketch. An infinite branch γ of $\mathsf{Foc}(\pi)$ may either be obtained by reversibility only after a certain point, or by alternating infinitely often synchronous and asynchronous phases. In the first case it is valid by Proposition 12 while in the latter case, Lemma 25 ensures the existence of a branch δ of π containing infinitely many positive rules of γ, with a valid thread t of minimal formula F_m: every rule r of δ in which F_m is principal is below a positive rule occurring in γ. Thus r occurs in γ, which is therefore valid. ◀

4 Cut elimination

In this section, we show that any $\mu\mathsf{MALL}^\infty$ proof can be transformed into an equivalent cut-free derivation. This is done by applying the cut reduction rules described in Section 2, possibly in infinite reductions converging to cut-free proofs. As usual with infinitary reductions it is not the case that any reduction sequence converges: for instance, one could reduce only deep cuts in a proof, leaving a cut untouched at the root. We avoid this problem by considering a form of head reduction where we only reduce cuts at the root.

Cut reduction rules are of two kinds, *principal* reductions and *auxiliary* ones. In the infinitary setting, principal cut reductions do not immediately contribute to producing a cut-free pre-proof. On the contrary, auxiliary cut reductions are productive in that sense. In other words, principal rules are seen as internal computations of the cut elimination process, while auxiliary rules are seen as a partial output of that process. Accordingly, the former will be called *internal rules* and the latter *external rules*.

When analyzing cut reductions, cut commutations can be troublesome. A common way to avoid this technicality [13], which we shall follow, is to introduce a *multicut* rule which merges multiple cuts, avoiding cut commutations.

$$\frac{s_1 \dots s_n}{s}\ (\mathsf{mcut})$$

▶ **Definition 27.** Given two sequents s and s', we say that they are cut-connected on a formula occurrence F when $F \in s$ and $F^\perp \in s'$. We say that they are cut-connected when they are cut-connected for some F. We define the *multicut* rule as shown above with

$$\dfrac{\dfrac{\vdash \Gamma, F \quad \vdash F^\perp, \Delta}{\vdash \Gamma, \Delta}\,(\mathsf{Cut}) \quad \dots}{\vdash \Sigma}\,(\mathsf{mcut}) \longrightarrow \dfrac{\vdash \Gamma, F \quad \vdash F^\perp, \Delta \quad \dots}{\vdash \Sigma}\,(\mathsf{mcut})$$

$$\dfrac{\dfrac{\vdash \Gamma, F}{\vdash \Gamma, F \oplus G} \quad \dfrac{\vdash G^\perp, \Delta \quad \vdash F^\perp, \Delta}{\vdash G^\perp \& F^\perp, \Delta} \quad \dots}{\vdash \Sigma}\,(\mathsf{mcut}) \longrightarrow \dfrac{\vdash \Gamma, F \quad \vdash F^\perp, \Delta \quad \dots}{\vdash \Sigma}\,(\mathsf{mcut})$$

$$\dfrac{s_1 \dots s_n \quad \dfrac{\vdash \Gamma, F \quad \vdash \Gamma, G}{\vdash \Gamma, F \& G}\,(\&)}{\vdash \Sigma, F \& G}\,(\mathsf{mcut}) \longrightarrow \dfrac{\dfrac{s_1 \dots s_n \quad \vdash \Gamma, F}{\vdash \Sigma, F}\,(\mathsf{mcut}) \quad \dfrac{s_1 \dots s_n \quad \vdash \Gamma, G}{\vdash \Sigma, G}\,(\mathsf{mcut})}{\vdash \Sigma, F \& G}\,(\&)$$

Figure 2 (Cut)/(mcut) and $(\oplus_1)$/(&) internal reductions and (&)/(mcut) external reduction.

conclusion s and premisses $\{s_i\}_i$, where the set $\{s_i\}_i$ is connected and acyclic with respect to the cut-connection relation, and s is the set of all formula occurrences F that appear in some s_i but such that no s_j is cut-connected to s_j on F.

From now on we shall work with $\mu\mathsf{MALL}^\infty_\mathsf{m}$ derivations, which are $\mu\mathsf{MALL}^\infty$ derivations in which the multicut rule may occur, though only at most once per branch. The notions of thread and validity are unchanged. In $\mu\mathsf{MALL}^\infty_\mathsf{m}$ we only reduce multicuts, in a way that is naturally obtained from the cut reductions of $\mu\mathsf{MALL}^\infty$. A complete description of the rules is given in [6]; only the (Cut)/(mcut) and $(\oplus_1)$/(&) internal reduction cases and the (&)/(mcut) external reduction case are shown in Figure 2. As is visible in the last reduction, applying an external rule on a multicut may yield multiple multicuts, though always on disjoint subtrees.

We will be interested in a particular kind of multicut reduction sequences, the *fair* ones, which are such that any redex which is available at some point of the sequence will eventually have disappeared from the sequence (being reduced or erased), details are provided in [6]. We will establish that these reductions eliminate multicuts:

▶ **Theorem 28.** *Fair multicut reduction sequences on* $\mu\mathsf{MALL}^\infty_\mathsf{m}$ *proofs produce* $\mu\mathsf{MALL}^\infty$ *proofs.*

Additionally, if all cuts in the initial derivation are above multicuts, the resulting $\mu\mathsf{MALL}^\infty$ derivation must actually be cut-free: indeed, multicut reductions never produce a cut. Thus Theorem 28 gives a way to eliminate cuts from any $\mu\mathsf{MALL}^\infty$ proof π of $\vdash \Gamma$ by forming a multicut with conclusion $\vdash \Gamma$ and π as unique subderivation, and eliminating multicuts (and cuts) from that $\mu\mathsf{MALL}^\infty_\mathsf{m}$ proof. The proof of Theorem 28 is in two parts. We first prove that fair internal multicut reductions cannot diverge (Proposition 40), hence fair multicut reductions are productive, *i.e.*, reductions of $\mu\mathsf{MALL}^\infty_\mathsf{m}$ proofs converge to $\mu\mathsf{MALL}^\infty$ pre-proofs. We then establish that the obtained pre-proof is a valid proof (Proposition 41).

Regarding productivity, assuming that there exists an infinite sequence σ of internal cut-reductions from a given proof π of Γ, we obtain a contradiction by extracting from π a proof of the empty sequent in a suitably defined proof-system. More specifically, we observe that no formula of Γ is principal in the subtree π_σ of π visited by σ. Hence, by erasing every formula of Γ from π_σ, local correctness of the proof is preserved, resulting in a tree deriving the empty sequent. This tree can be viewed as a proof in a new proof-system $\mu\mathsf{MALL}^\infty_\tau$ which is shown to be sound (Proposition 37) with respect to the traditional Boolean semantics of the μ-calculus, thus the contradiction. The proof of validity of the produced pre-proof is

similar: instead of extracting a proof of the empty sequent from π we will extract, for each invalid branch of π, a $\mu\mathsf{MALL}^{\infty}_{\tau}$ proof of a formula containing neither $\mathbf{1}$, $\top$, nor ν formulas, contradicting soundness again.

4.1 Extracting proofs from reduction paths

We define now a key notion to analyze the behaviour of multicut-elimination: given a multicut reduction starting from π, we extract a (slightly modified) subderivation of π which corresponds to the part of the derivation that has been explored by the reduction. More precisely, we are interested in *reduction paths* which are sequences of proofs that end with a multicut rule, obtained by tracing one multicut through its evolution, selecting only one sibling in the case of $(\&)$ and $(\otimes)$ external reductions. Given such a reduction path starting with π, we consider the subtree of π whose sequents occur in the reduction path as premises of some multicut. This subtree is obviously not always a $\mu\mathsf{MALL}^{\infty}$ derivation since some of its nodes may have missing premises. We will provide an extension of $\mu\mathsf{MALL}^{\infty}$ where these trees can be viewed as proper derivations by first characterizing when this situation arises.

▶ **Definition 29** (Useless sequents, distinguished formula)**.** Let $\mathcal{R}$ be a reduction path starting with π. A sequent $s = (\vdash \Gamma, F)$ of π is said to be *useless* with *distinguished formula* F when in one of the following cases:

1. The sequent eventually occurs as a premise of all multicuts of $\mathcal{R}$ and F is the principal formula of s in π. (Note that the distinguished formula F of a useless sequent s of sort (1) must be a sub-occurrence of a cut formula in π. Otherwise, the fair reduction path $\mathcal{R}$ would eventually have applied an external rule on s. Moreover, $F^{\perp}$ never becomes principal in the reduction path, otherwise by fairness the internal rule reducing F and $F^{\perp}$ would have been applied.)
2. At some point in the reduction, the sequent is a premise of $(\&)$ on $F\&F'$ or $F'\&F$ which is erased in an internal $(\&)/(\oplus)$ multicut reduction. (In the $(\oplus_1)/(\&)$ internal reduction of Figure 2, the sequent $\vdash G^{\perp}, \Delta$ is useless of sort (2).)
3. The sequent is ignored at some point in the reduction path because it is not present in the selected multicut after a branching external reduction on $F \star F'$ or $F' \star F$, for $\star \in \{\otimes, \&\}$. (In the $(\&)/(\mathsf{mcut})$ external reduction of Figure 2, if one is considering a reduction path that follows the multicut having $\vdash \Gamma, F$ as a premise, then the sequent $\vdash \Gamma, G$ is useless of sort (3), and vice versa.)
4. The sequent is ignored at some point in the reduction path because a $(\otimes)/(\mathsf{mcut})$ external reduction distributes s to the multicut that is not selected in the path. This case will be illustrated next, and is described in full details in [6].

Note that, although the external reduction for $\top$ erases sequents, we do not need to consider such sequents as useless: indeed, we will only need to work with useless sequents in infinite reduction paths, and the external reduction associated to $\top$ terminates a path.

▶ **Example 30.** Consider a multicut composed of the last example of Section 2 and an arbitrary proof of $\vdash F, \Delta$ where F is principal. In the reduction paths which always select the right premise of an external $(\otimes)/(\mathsf{mcut})$ corresponding to the $N'\otimes S'$ formulas, the sequent $\vdash F, \Delta$ will always be present and thus useless by case (1). In the reduction paths which eventually select a left premise, the sequent $N_2, F \vdash S'$ is useless of sort (3) with S' distinguished, and $\vdash F, \Delta$ is useless of sort (4) with F distinguished.

In order to obtain a proper pre-proof from the sequents occurring in a reduction path, we need to close the derivation on useless sequents. This is done by replacing distinguished

formulas by $\top$ formulas. However, a usual substitution is not appropriate here as we are really replacing formula occurrence, which may be distributed in arbitrarily complex ways among sub-occurrences.

▶ **Definition 31.** A *truncation* τ is a partial function from Σ^* to $\{\top, \mathbf{0}\}$ such that:
- For any $\alpha \in \Sigma^*$, if $\alpha \in \mathtt{Dom}(\tau)$, then $\alpha^\perp \in \mathtt{Dom}(\tau)$ and $\tau(\alpha) = \tau(\alpha^\perp)^\perp$.
- If $\alpha \in \mathtt{Dom}(\tau)$ then for any $\beta \in \Sigma^+$, $\alpha.\beta \notin \mathtt{Dom}(\tau)$.

▶ **Definition 32** (Truncation of a reduction path)**.** Let $\mathcal{R}$ be a reduction path. The truncation τ associated to $\mathcal{R}$ is defined by setting $\tau(\alpha) = \top$ and $\tau(\alpha^\perp) = \mathbf{0}$ for every formula occurrence φ_α that is distinguished in some useless sequent of $\mathcal{R}$.

The above definition is justified because F and $F^\perp$ cannot both be distinguished, by fairness of $\mathcal{R}$. We can finally obtain the pre-proof associated to a reduction path, in a proof system slightly modified to take truncations into account.

▶ **Definition 33** (Truncated proof system)**.** Given a truncation τ, the infinitary proof system $\mu\mathsf{MALL}^\infty_\tau$ is obtained by taking all the rules of $\mu\mathsf{MALL}^\infty$, with the proviso that they only apply when the address of their principal formula is not in the domain of τ, with the following extra rule:

$$\frac{\vdash \tau(\alpha)_{\alpha i}, \Delta}{\vdash F, \Delta}\ (\tau) \quad \text{if } \alpha \in \mathtt{Dom}(\tau)$$

The address $\alpha.i$ associated with $\tau(\alpha)$ in the rule (τ) forbids loops on a (τ) rule. Indeed if $\alpha \in \mathtt{Dom}(\tau)$ then $\alpha.i \notin \mathtt{Dom}(\tau)$.

▶ **Definition 34** (Truncated proof associated to a reduction path)**.** Let $\mathcal{R}$ be a fair infinite reduction path starting with π and τ be the truncation associated to it. We define $TR(\mathcal{R})$ to be the $\mu\mathsf{MALL}^\infty_\tau$ proof obtained from π by keeping only sequents that occur as premise of some multicut in $\mathcal{R}$, using the same rules as in π whenever possible, and deriving useless sequents by rules (τ) and $(\top)$.

This definition is justified by definition of τ and because only useless sequents may be selected without their premises (in π) being also selected. Notice that the dual $F^\perp$ of a distinguished formula F may only occur in $\mathcal{R}$ for distinguished formulas of type (1) and (4); in these cases $F^\perp$ is never principal in $\mathcal{R}$ by fairness. Thus, there is no difficulty in constructing $TR(\mathcal{R})$ with a truncation defined on the address of $F^\perp$. Finally, note that $TR(\mathcal{R})$ is indeed a valid $\mu\mathsf{MALL}^\infty_\tau$ pre-proof, because its infinite branches are infinite branches of π.

▶ **Example 35.** Continuing the previous example, we consider the path where the left premise of the tensor is selected immediately. The associated truncation is such that $\tau(S') = \top$ and $\tau(F) = \top$ by (3) and (4) respectively. The derivation $TR(\mathcal{R})$ is shown below, where Π_{ax} denotes the expansion of the axiom given by Proposition 10:

$$\dfrac{\dfrac{}{\vdash F, \Delta}\ (\tau),(\top) \qquad \dfrac{\dfrac{\dfrac{\Pi_{\mathsf{dup}}}{N \vdash N_1 \otimes N_2} \qquad \dfrac{\dfrac{\dfrac{\Pi_{\mathsf{ax}}}{N_1 \vdash N'} \qquad \dfrac{}{N_2, F \vdash S'}\ (\tau),(\top)}{N_1, N_2, F \vdash N' \otimes S'}}{N_1 \otimes N_2, F \vdash N' \otimes S'}\ (\mathsf{Cut})}{N, F \vdash N' \otimes S'}}{N, F \vdash S}\ (\mathsf{mcut})}{N \vdash S, \Delta}$$

4.2 Truncated truth semantics

We fix a truncation τ and define a truth semantics with respect to which $\mu\mathsf{MALL}^\infty_\tau$ will be sound. The semantics is classical, assigning a Boolean value to formula occurrences. For convenience, we take $\mathcal{B} = \{\mathbf{0}, \top\}$ as our Boolean lattice, with $\wedge$ and $\vee$ being the usual meet and join operations on it. The following definition provides an interpretation of $\mu\mathsf{MALL}$ formulas which consists in the composition of the standard interpretation of μ-calculus formulas with the obvious linearity-forgetting translation from $\mu\mathsf{MALL}$ to classical μ-calculus.

▶ **Definition 36.** Let φ_α be a pre-formula occurrence. We call *environment* any function $\mathcal{E}$ mapping free variables of φ to (total) functions of $E := \Sigma^* \to \mathcal{B}$. We define $[\varphi_\alpha]^\mathcal{E} \in \mathcal{B}$, the *interpretation* of φ_α in the environment $\mathcal{E}$, by $[\varphi_\alpha]^\mathcal{E} = \tau(\alpha)$ if $\alpha \in \mathtt{Dom}(\tau)$, and otherwise:

- $[X_\alpha]^\mathcal{E} = \mathcal{E}(X)(\alpha)$, $[\top_\alpha]^\mathcal{E} = [\mathbf{1}_\alpha]^\mathcal{E} = \top$ and $[\mathbf{0}_\alpha]^\mathcal{E} = [\bot_\alpha]^\mathcal{E} = \mathbf{0}$.
- $[(\varphi \mathbin{⍉} \psi)_\alpha]^\mathcal{E} = [\varphi_{\alpha.l}]^\mathcal{E} \wedge [\psi_{\alpha.r}]^\mathcal{E}$, for $⍉ \in \{\&, \otimes\}$.
- $[(\varphi \mathbin{⍊} \psi)_\alpha]^\mathcal{E} = [\varphi_{\alpha.l}]^\mathcal{E} \vee [\psi_{\alpha.r}]^\mathcal{E}$, for $⍊ \in \{\oplus, ⅋\}$.
- $[(\mu X.\varphi)_\alpha]^\mathcal{E} = \mathrm{lfp}(f)(\alpha)$ and $[(\nu X.\varphi)_\alpha]^\mathcal{E} = \mathrm{gfp}(f)(\alpha)$ where $f : E \to E$ is given by $f : h \mapsto \beta \mapsto \big(\tau(\beta)$ if $\beta \in \mathtt{Dom}(\tau)$ and $[\varphi_{\beta.i}]^{\mathcal{E}::X\mapsto h}$ otherwise$\big)$.

When F is closed, we simply write $[F]$ for $[F]^\emptyset$.

We refer the reader to the long version [6] for details on the construction of the interpretation. We simply state here the main result about it.

▶ **Proposition 37.** *If $\vdash \Gamma$ is provable in $\mu\mathsf{MALL}^\infty_\tau$, then $[F] = \top$ for some $F \in \Gamma$.*

We only sketch the soundness proof (see [6] for details) which proceeds by contradiction. Assuming we are given a proof π of a formula F such that $[F] = \mathbf{0}$, we exhibit a branch β of π containing only formulas interpreted by $\mathbf{0}$. A validating thread of β unfolds infinitely often some formula $\nu X.\varphi$. Since the interpretation of $\nu X.\varphi$ is defined as the gfp of a monotonic operator f we have, for each occurrence $(\nu X.\varphi)_\alpha$ in β, an ordinal λ such that $[(\nu X.\varphi)_\alpha] = f^\lambda(\bigvee E)(\alpha)$, where $\bigvee E$ is the supremum of the complete lattice E. We show that this ordinal can be forced to decrease along β at each fixed point unfolding, contradicting the well-foundedness of the class of ordinals.

▶ **Definition 38.** A truncation τ is *compatible* with a formula φ_α if $\alpha \notin dom(\tau)$ and, for any $\alpha \sqsubseteq \beta.d \in \mathtt{Dom}(\tau)$ where $d \in \{l, r, i\}$, we have that φ_α admits a sub-occurrence ψ_β with $\otimes$ or $\&$ as the toplevel connective of ψ, $d \in \{l, r\}$, and $\alpha.d' \notin \mathtt{Dom}(\tau)$ for any $d \neq d'$.

In other words, a truncation τ is compatible with a formula F if it truncates only sons of $\otimes$ or $\&$ nodes in the tree of the formula F and at most one son of each such node.

▶ **Proposition 39.** *If F is a formula compatible with τ and containing no ν binders, no $\top$ and no $\mathbf{1}$, then $[F] = \mathbf{0}$.*

4.3 Proof of cut elimination

We first show that multicut reduction is productive, then that the resulting (cut-free) pre-proof is actually a valid proof.

▶ **Proposition 40.** *Any fair reduction sequence produces a $\mu\mathsf{MALL}^\infty$ pre-proof.*

Proof. By contradiction, consider a fair infinite sequence of internal multicut reductions. This sequence is a fair reduction path $\mathcal{R}$. Let τ and $TR(\mathcal{R})$ be the associated truncations and truncated proof. Since no external reduction occurs, it means that conclusion formulas of $TR(\mathcal{R})$ are never principal in the proof, thus we can transform it into a proof of the empty sequent, which contradicts soundness of $\mu\mathsf{MALL}^\infty_\tau$. ◀

▶ **Proposition 41.** *Any fair mcut-reduction produces a* $\mu\mathsf{MALL}^\infty$ *proof.*

Proof. Let π be a $\mu\mathsf{MALL}^\infty_{\mathsf{m}}$ proof of conclusion $\vdash \Gamma$, and π' the cut-free pre-proof obtained by Proposition 40, *i.e.*, the limit of the multicut reduction process. Any branch of π' corresponds to a multicut reduction path. For the sake of contradiction, assume that π' is invalid. It must thus have an invalid infinite branch, corresponding to an infinite reduction path $\mathcal{R}$. Let τ and $\theta := TR(\mathcal{R})$ be the associated truncation and truncated proof in $\mu\mathsf{MALL}^\infty_\tau$.

We first observe that formulas of Γ cannot have suboccurrences of the form $\mathbf{1}_\alpha$ or $\top_\alpha$ that are principal in π'. Indeed, this could only be produced by an external rule $(\top)/(\mathsf{mcut})$ in the reduction path $\mathcal{R}$, but that would terminate the path, contradicting its infiniteness.

Next, we claim that all threads starting from formulas in Γ are invalid. Indeed, all rules applied to those formulas are transferred (by means of external rules) to the branch produced by the reduction path. The existence of a valid thread starting from the conclusion sequent in θ would thus imply the existence of a valid thread in our branch of π'.

By the first observation, we can replace all $\mathbf{1}$ and $\top$ subformulas of Γ by $\mathbf{0}$ without changing the derivation, and obviously without breaking its validity. By the second observation, we can further modify Γ by changing all ν combinators into μ combinators. The derivation is easily adapted (using rule (μ) instead of (ν)) and it remains valid, since the validity of θ could not have been caused by a valid thread starting from the root. We thus obtain a valid pre-proof θ' of $\vdash \Gamma'$ in $\mu\mathsf{MALL}^\infty_\tau$, where Γ' contains no ν, $\mathbf{1}$ and $\top$.

We finally show that τ is compatible with any formula occurrence from Γ. Indeed, if $\tau(\beta)$ is defined for some suboccurrence ψ_β of a formula $\varphi_\alpha \in \Gamma$, then it can only be because of a useless sequent of sort (3), *i.e.*, a truncation due to the fact that the reduction path has selected only one sibling after a branching external rule. We thus conclude, by Proposition 39, that all formulas of Γ are interpreted as $\mathbf{0}$ in the truncated semantics associated to τ, which contradicts the validity of θ' and Proposition 37. ◀

5 Conclusion

We have established focalization and cut elimination for $\mu\mathsf{MALL}^\infty$, the infinitary sequent calculus corresponding to $\mu\mathsf{MALL}$. Our cut elimination result extends that of Santocanale and Fortier [13], but this extension has required the elaboration of a radically different proof technique.

An obvious direction for future work is now to go beyond linear logic, and notably handle structural rules in infinitary cut elimination. But many interesting questions are also left in the linear case. First, it will be natural to relax the hypothesis on fairness in the cut-elimination result. Other than cut elimination, the other long standing problem regarding $\mu\mathsf{MALL}^\infty$ and similar proof systems is whether regular proofs can be translated, in general, to finitary proofs. Further, one can ask the same question, requiring in addition that the computational content of proofs is preserved in the translation. It may well be that regular $\mu\mathsf{MALL}^\infty$ contains more computations than $\mu\mathsf{MALL}$; even more so if one considers other classes of finitely representable infinitary proofs. It would be interesting to study how this could impact the study of programming languages for (co)recursion, and understanding links with other approaches to this question [1, 2]. In this direction, we will be interested in studying the computational interpretation of focused cut-elimination, providing a logical basis for inductive and coinductive matching in regular and infinitary proof systems.

References

1 Andreas Abel and Brigitte Pientka. Wellfounded recursion with copatterns: a unified approach to termination and productivity. In Greg Morrisett and Tarmo Uustalu, editors, *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA – September 25-27, 2013*, pages 185–196. ACM, 2013. doi:10.1145/2500365.2500591.

2 Andreas Abel, Brigitte Pientka, David Thibodeau, and Anton Setzer. Copatterns: programming infinite structures by observations. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL'13, Rome, Italy – January 23-25, 2013*, pages 27–38. ACM, 2013. doi:10.1145/2429069.2429075.

3 Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

4 David Baelde. Least and greatest fixed points in linear logic. *ACM Transactions on Computational Logic (TOCL)*, 13(1):2, 2012.

5 David Baelde, Amina Doumane, and Alexis Saurin. Least and greatest fixed points in ludics. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7-10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 549–566. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.549.

6 David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory : the multiplicative additive case. *HAL*, hal-01339037, June 2016. URL: https://hal.archives-ouvertes.fr/hal-01339037.

7 Michele Basaldella, Alexis Saurin, and Kazushige Terui. On the meaning of focalization. In Alain Lecomte and Samuel Tronçon, editors, *Ludics, Dialogue and Interaction – PRELUDE Project – 2006-2009. Revised Selected Papers*, volume 6505 of *Lecture Notes in Computer Science*, pages 78–87. Springer, 2011. doi:10.1007/978-3-642-19211-1_5.

8 James Brotherston and Nikos Gorogiannis. Cyclic abduction of inductively defined safety and termination preconditions. In Markus Müller-Olm and Helmut Seidl, editors, *Static Analysis – 21st International Symposium, SAS 2014, Munich, Germany, September 11-13, 2014. Proceedings*, volume 8723 of *Lecture Notes in Computer Science*, pages 68–84. Springer, 2014. doi:10.1007/978-3-319-10936-7_5.

9 James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, December 2011.

10 Pierre-Louis Curien and Guillaume Munch-Maccagnoni. The duality of computation under focus. In Cristian S. Calude and Vladimiro 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 165–181. Springer, 2010. doi:10.1007/978-3-642-15240-5_13.

11 Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time μ-calculus. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, pages 273–284, 2006. doi:10.1007/11944836_26.

12 Amina Doumane, David Baelde, Lucca Hirschi, and Alexis Saurin. Towards Completeness via Proof Search in the Linear Time mu-Calculus. Accepted for publication at LICS, January 2016. URL: https://hal.archives-ouvertes.fr/hal-01275289.

13 Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL*

2013), CSL 2013, September 2-5, 2013, Torino, Italy, volume 23 of *LIPIcs*, pages 248–262. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2013. doi:10.4230/LIPIcs.CSL.2013.248.

14 David Janin and Igor Walukiewicz. Automata for the modal mu-calculus and related results. In Jirí Wiedermann and Petr Hájek, editors, *Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS'95, Prague, Czech Republic, August 28 – September 1, 1995, Proceedings*, volume 969 of *Lecture Notes in Computer Science*, pages 552–562. Springer, 1995. doi:10.1007/3-540-60246-1_160.

15 Roope Kaivola. Axiomatising linear time mu-calculus. In *CONCUR'95: Concurrency Theory, 6th International Conference, Philadelphia, PA, USA, August 21-24, 1995, Proceedings*, pages 423–437, 1995. doi:10.1007/3-540-60218-6_32.

16 Dexter Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

17 Olivier Laurent. Polarized games. *Annals of Pure and Applied Logic*, 130(1-3):79–123, 2004. doi:10.1016/j.apal.2004.04.006.

18 Olivier Laurent. A proof of the focalization property of linear logic. Unpublished note, May 2004.

19 Paul-André Melliès and Nicolas Tabareau. Resource modalities in tensor logic. *Annals of Pure and Applied Logic*, 161(5):632–653, 2010. doi:10.1016/j.apal.2009.07.018.

20 Dale Miller and Alexis Saurin. From proofs to focused proofs: A modular proof of focalization in linear logic. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 405–419. Springer, 2007. doi:10.1007/978-3-540-74915-8_31.

21 Luigi Santocanale. μ-bicomplete categories and parity games. *ITA*, 36(2):195–227, 2002. doi:10.1051/ita:2002010.

22 Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures*, volume 2303 of *Lecture Notes in Computer Science*, pages 357–371. Springer, 2002. doi:10.1007/3-540-45931-6_25.

23 Luigi Santocanale. Free μ-lattices. *Journal of Pure and Applied Algebra*, 168(2–3):227–264, March 2002. URL: https://hal.archives-ouvertes.fr/hal-01261049, doi:10.1016/S0022-4049(01)00098-6.

24 Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, 1989. doi:10.1016/0890-5401(89)90031-X.

25 Kazushige Terui. Computational ludics. *Theoretical Computer Science*, 412(20):2048–2071, 2011. doi:10.1016/j.tcs.2010.12.026.

26 Igor Walukiewicz. On completeness of the mu-calculus. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS'93), Montreal, Canada, June 19-23, 1993*, pages 136–146. IEEE Computer Society, 1993. doi:10.1109/LICS.1993.287593.

27 Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional mu-calculus. In *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pages 14–24. IEEE Computer Society, 1995. doi:10.1109/LICS.1995.523240.

28 Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional mu-calculus. *Information and Computation*, 157(1-2):142–182, 2000. doi:10.1006/inco.1999.2836.

29 Noam Zeilberger. *The logical basis of evaluation order and pattern matching*. PhD thesis, The logical basis of evaluation order and pattern matching, 2009.