Lev D. Beklemishev
Ruy de Queiroz (Eds.)

LNAI 6642

Logic, Language, Information and Computation

18th International Workshop, WoLLIC 2011
Philadelphia, PA, USA, May 2011
Proceedings

Springer

Lecture Notes in Artificial Intelligence 6642

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

FoLLI Publications on Logic, Language and Information

Lev D. Beklemishev   Ruy de Queiroz (Eds.)

Logic, Language, Information and Computation

18th International Workshop, WoLLIC 2011
Philadelphia, PA, USA, May 18-20, 2011
Proceedings

Springer

Volume Editors

Lev D. Beklemishev
Steklov Mathematical Institute
Gubkina 8, 119991 Moscow, Russia
E-mail: bekl@mi.ras.ru

Ruy de Queiroz
Universidade Federal de Pernambuco, Centro de Informática
Avenida Prof. Luis Freire s/n, Cidade Universitária
50740-540 Recife, PE, Brazil
E-mail: ruy@cin.ufpe.br

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20919-2 e-ISBN 978-3-642-20920-8
DOI 10.1007/978-3-642-20920-8
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011926457

CR Subject Classification (1998): F.4.1, F.3, F.4, I.2.3, G.2, I.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at WoLLIC 2011: 18th Workshop on Logic, Language, Information and Computation, held during May 18–20, 2011, at the University of Pennsylvania, Philadelphia, USA.

WoLLIC is a series of workshops which started in 1994 with the aim of fostering interdisciplinary research in pure and applied logic. The idea was to have a forum which is large enough in the number of possible interactions between logic and the sciences related to information and computation, and yet is small enough to allow for concrete and useful interaction among participants.

WoLLIC 2011 included invited lectures by Rajeev Alur (Philadelphia), Rosalie Iemhoff (Utrecht), John Mitchell (Stanford), Vladimir Voevodsky (Princeton), Yoad Winter (Utrecht), and Michael Zakharyaschev (London).

There were 35 submissions, of which the committee decided to accept 21 papers. In addition, WoLLIC 2011 had a special session dedicated to the 65th birthday of Max Kanovich. This session was organized by Andre Scedrov.

The work of the Program Committee was greatly facilitated by the use of the EasyChair conference management system created by Andrei Voronkov.

We should also like to thank the members of the Program Committee as well as the external reviewers for keeping the scientific standards high. The work of the Organizing Committee, represented by Andre Scedrov (Local Chair) and Monica Pallanti (senior departmental administrator, Department of Mathematics, University of Pennsylvania), is greatly appreciated.

Scientific sponsorship of WoLLIC over the years has consistently come from the Interest Group in Pure and Applied Logics (IGPL), the The Association for Logic, Language and Information (FoLLI), the Association for Symbolic Logic (ASL), the European Association for Theoretical Computer Science (EATCS), the European Association for Computer Science Logic (EACSL), the Sociedade Brasileira de Computação (SBC), and the Sociedade Brasileira de Lógica (SBL).

We acknowledge substantial help from Evgeny Dashkov in producing this volume.

March 2011

Lev Beklemishev
Ruy De Queiroz

Organization

Program Committee

Sergei Artemov	CUNY Graduate Center, New York, USA
Jeremy Avigad	Carnegie Mellon University, Pittsburgh, USA
Arnold Beckmann	Swansea University, UK
Lev Beklemishev (Chair)	Steklov Institute of Mathematics, Moscow, Russia
Alessandro Berarducci	University of Pisa, Italy
Sam Buss	University of California, San Diego, USA
Ruy De Queiroz	Universidade Federal de Pernambuco, Brazil
Achim Jung	University of Birmingham, UK
Benedikt Löwe	University of Amsterdam and University of Hamburg, The Netherlands and Germany
Johann Makowski	Technion, Haifa, Israel
Michael Moortgat	Utrecht University, The Netherlands
Prakash Panangaden	McGill University, Montreal, Canada
Rohit Parikh	CUNY Graduate Center and Brooklyn College, New York, USA
Alexander Shen	LIF CNRS, Marseille and IITP RAS, Moscow, France and Russia
Bas Spitters	Radboud University Nijmegen, The Netherlands
Vincent Van Oostrom	Utrecht University, The Netherlands
Helmut Veith	Vienna University of Technology, Austria
Yde Venema	University of Amsterdam, The Netherlands
Scott Weinstein	University of Pennsylvania, USA
Frank Wolter	University of Liverpool, UK

Steering Committee

Samson Abramsky
Johan van Benthem
Anuj Dawar
Joe Halpern
Wilfrid Hodges
Daniel Leivant
Angus Macintyre
Grigori Mints
Hiroakira Ono
Ruy de Queiroz

Organizing Committee

Vivek Nigam
Anjolina G. de Oliveira
Ruy de Queiroz (Co-chair)
Andre Scedrov (Co-chair)

External Reviewers

A
Arai, Toshiyasu
Areces, Carlos
B
Bezhanishvili, Nick
Bradfield, Julian
Bronnikov, George
C
Corradini, Andrea
D
Dechesne, Francien
Dekkers, Wil
Dezani, Mariangiola
F
Fiorino, Guido
Fischer, Eldar
Franco, Giuditta
G
Gabbay, Murdoch
Ganzow, Tobias
Geurts, Bart
Guillon, Pierre
H
Hindley, Roger
Hodkinson, Ian
J
Johanson, Lars
K
Kanza, Yaron
Katz, Shmuel
Krivelevich, Michael
Kupke, Clemens
Kurz, Alexander
Kuznets, Roman
L
Lengye, Floria
Lim, Dongsik
M
Manzonetto, Giulio
McCready, Eric
Mera, Sergio
Milnikel, Bob
Moller, Faron
O
Odintsov, Sergei
P
Philip, Bill
Piro, Robert
R
Rosicky, Jiri
S
Salibra, Antonino
Schneider, Thomas
Schröder, Bernhard
Simon, Sunil
Sorbi, Andrea
Steedman, Mark
Sutner, Klaus
T
Troquard, Nicolas
Tzevelekos, Nikos
V
Van Eijck, Jan
Verbrugge, Rineke
W
Witzel, Andreas
Z
Zolin, Evgeny
Zvesper, Jonathan

Table of Contents

Section 1: Invited Talks

Section 2: Contributed Papers

Streaming String Transducers

Rajeev Alur

Department of Computer and Information Science,
University of Pennsylvania
alur@cis.upenn.edu

Streaming string transducers define (partial) functions from input strings to output strings. A streaming string transducer makes a single pass through the input string and uses a finite set of variables that range over strings from the output alphabet. At every step, the transducer processes an input symbol, and updates all the variables in parallel using assignments whose right-hand-sides are concatenations of output symbols and variables with the restriction that a variable can be used at most once in a right-hand-side expression. The expressiveness of streaming string transducers coincides with the class of "regular" transductions that can be equivalently defined using two-way deterministic finite-state transducers and using monadic second-order logic. The problems of checking functional equivalence of two streaming transducers, and of checking whether a streaming transducer satisfies pre/post verification conditions specified by finite automata, are solvable in PSPACE. These decision procedures also generalize to the model of streaming transducers over *data strings*—strings over symbols tagged with data values over a potentially infinite data domain that supports only the operations of equality and ordering. We identify a class of imperative and a class of functional programs, manipulating lists of data items, which can be effectively translated to such streaming data-string transducers. Our results lead to algorithms for assertion checking and for checking functional equivalence of two programs, written possibly in different programming styles, for commonly used routines such as insert, delete, and reverse.

This talk is based on results reported in [2] and [1].

References

1. Alur, R., Černý, P.: Expressiveness of streaming string transducers. In: Proc. 30th FSTTCS, pp. 1–12 (2010)
2. Alur, R., Černý, P.: Streaming transducers for algorithmic verification of single-pass list-processing programs. In: Proc. 38th POPL, pp. 599–610 (2011)

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, p. 1, 2011.

Unification in Logic

Rosalie Iemhoff

Department of Philosophy,
Utrecht University
Rosalie.Iemhoff@phil.uu.nl

There are many problems in mathematics that can be cast in terms of unification, meaning that a solution of the problem is a substitution that identifies two terms, either literally, or against a background theory of equivalence. In the context of logics, a unifier is a substitution under which a formula becomes derivable in the logic.

In classical propositional logic, all unifiable formulas have a *most general unifier*, which is a unifier that generates all other unifiers of a formula. Nonclassical logics in general do not have this useful property, but many modal and intermediate propositional logics satisfy a slightly weaker property. In these logics, for every formula there is a finite set of unifiers such that any other unifier of the formula is generated by one of them.

The study of unification in nonclassical logics mainly uses semantical techniques. Even though there exist algorithms to find unifiers, proofs of correctness again use semantics. In this talk a purely syntactic treatment of unification is presented, and it is shown that most known results can be obtained, and in some cases extended, by this approach.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, p. 2, 2011.

A Symbolic Logic with Exact Bounds for Cryptographic Protocols

John C. Mitchell

Department of Computer Science,
Stanford University
mitchell@cs.stanford.edu

This invited talk will describe a formal logic for reasoning about security properties of network protocols with proof rules indicating exact security bounds that could be used to choose key lengths or other concrete security parameters. The soundness proof for this logic, a variant of previous versions of Protocol Composition Logic (PCL), shows that derivable properties are guaranteed in a standard cryptographic model of protocol execution and resource-bounded attack. We will discuss the general system and present example axioms for digital signatures and random nonces, with concrete security properties based on concrete security of signature schemes and pseudorandom number generators (PRG). The quantitative formal logic supports first-order reasoning and reasoning about protocol invariants, taking exact security bounds into account. Proofs constructed in this logic also provide conventional asymptotic security guarantees because of the way that exact bounds accumulate in proofs. As an illustrative example producing exact bounds, we use the formal logic to prove an authentication property with exact bounds of a signature-based challenge-response protocol.

This talk presents joint work with Anupam Datta (Carnegie Mellon University), Joseph Y. Halpern (Cornell University), and Arnab Roy (IBM Thomas J. Watson Research Center).

Acknowledgements. This work was partially supported by the National Science Foundation, the Air Force Office of Scientific Research, and the Office of Naval Research.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, p. 3, 2011.

Univalent Foundations of Mathematics

Vladimir Voevodsky

School of Mathematics,
Institute for Advanced Study
vladimir@math.ias.edu

Over the last two years deep and unexpected connections have been discovered between constructive type theories and classical homotopy theory. These connections open a way to construct new foundations of mathematics alternative to the ZFC. These foundations promise to resolve several seemingly unconnected problems—provide a support for categorical and higher categorical arguments directly on the level of the language, make formalizations of usual mathematics much more concise and much better adapted to the use with existing proof assistants such as Coq and finally to provide a uniform way to approach constructive and "classical" mathematics. I will try to describe the basic construction of a model of constructive type theories which underlies these innovations and provide some demonstration on how this model is used to develop mathematics in Coq.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, p. 4, 2011.

Relational Concepts and the Logic of Reciprocity

Yoad Winter

Department of Modern Languages and
Utrecht Institute of Linguistics OTS,
Utrecht University
y.winter@uu.nl

Research in logical semantics of natural language has extensively studied the functions denoted by expressions like *each other*, *one another* or *mutually.* The common analysis takes such **reciprocal expressions** to denote $\langle 1, 2\rangle$ generalized quantifiers over a given domain E, i.e. relations between subsets of E and binary relations over E. One of the reoccurring problems has been that reciprocal expressions seem to denote different quantifiers in different sentences. For instance, the reciprocal expression *each other* means the different quantifiers Q_1 and $Q_2 \subseteq \wp(E) \times \wp(E^2)$ in sentences (1) and (2), respectively.

(1) The girls know each other.
$Q_1 = \{\langle A, R\rangle : |A| \geq 2 \ \wedge \ \forall x, y \in A\, [x \neq y \rightarrow R(x, y)]\, \}$

(2) The girls are standing on each other.
$Q_2 = \{\langle A, R\rangle : |A| \geq 2 \ \wedge \ \forall x \in A\, \exists y \in A\, [x \neq y \wedge (R(x, y) \vee R(y, x))]\, \}$

Following previous work on reciprocals [1,2,3,4] I will suggest that the quantifier that a reciprocal expression denotes takes as parameter certain logical/cognitive semantic properties of **relational concepts** – intensions of two place predicates. In simple cases this parametrization only involves familiar properties like asymmetry or acyclicity, cf. the relation *stand on* in (2). However, in more complex cases, also preferences of use and other contextual parameters should be formally described as affecting the logical semantics of reciprocity. The precise formalization of such parameters and their affects on the selection of reciprocal quantifiers will be analyzed, based on recent joint work, logical as well as experimental, with Nir Kerem, Eva Poortman, Sivan Sabato and Naama Friedmann.

References

1. Dalrymple, M., Kanazawa, M., Kim, Y., Mchombo, S., Peters, S.: Reciprocal expressions and the concept of reciprocity. Linguistics and Philosophy 21, 159–210 (1998)
2. Kerem, N., Friedmann, N., Winter, Y.: Typicality effects and the logic of reciprocity. In: Proceedings of SALT19 (Semantics and Linguistic Theory) (to appear)
3. Sabato, S., Winter, Y.: From semantic restrictions to reciprocal meanings. In: Proceedings of FG-MOL (2005)
4. Winter, Y.: Plural predication and the Strongest Meaning Hypothesis. Journal of Semantics 18, 333–365 (2001)

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, p. 5, 2011.

Logic in the Time of WWW: An OWL View

Michael Zakharyaschev

Department of Computer Science and Information Systems,
Birkbeck College London, U.K.
michael@dcs.bbk.ac.uk

This paper analyses the classical automated reasoning problem—given a theory $\mathcal{T}$, a set $\mathcal{A}$ of ground atoms and a formula φ, decide whether $(\mathcal{T}, \mathcal{A}) \models \varphi$—in the context of the *OWL 2* Web Ontology Language and ontology-based data access (OBDA). In a typical OBDA scenario, $\mathcal{T}$ is an *OWL 2* 'ontology' providing a user-oriented view of raw data $\mathcal{A}$, and $\varphi(\boldsymbol{x})$ is a query with answer variables $\boldsymbol{x}$. Unlike classical automated reasoning, an important requirement for OBDA is that it should scale to large amounts of data and preferably be as efficient as standard relational database management systems. There are various ways of formalising this requirement, which give rise to different fragments of first-order logic suitable for OBDA. For example, according to the *query-rewriting approach* of [1], given $\mathcal{T}$, $\mathcal{A}$ and $\varphi(\boldsymbol{x})$, one has to compute a new query $\varphi'(\boldsymbol{x})$, independently of $\mathcal{A}$, such that, for any tuple $\boldsymbol{a}$, $(\mathcal{T}, \mathcal{A}) \models \varphi(\boldsymbol{a})$ iff $\mathcal{A} \models \varphi'(\boldsymbol{a})$. As a result, this approach can only be applicable to the languages for which query-answering belongs to the class AC^0 for *data complexity* (that is, if only $\mathcal{A}$ is regarded as input, whereas both $\mathcal{T}$ and φ are regarded as fixed). In the *combined approach* to OBDA [4,2], given $\mathcal{T}$, $\mathcal{A}$ and $\varphi(\boldsymbol{x})$, one has to compute new (i) $\mathcal{A}' \supseteq \mathcal{A}$ in polynomial time in $\mathcal{T}$ and $\mathcal{A}$, and (ii) $\varphi'(\boldsymbol{x})$ in polynomial tome in $\mathcal{T}$ and φ such that, for any tuple $\boldsymbol{a}$, $(\mathcal{T}, \mathcal{A}) \models \varphi(\boldsymbol{a})$ iff $\mathcal{A}' \models \varphi'(\boldsymbol{a})$. The combined approach can be used for languages with polynomial query-answering.

One focus of this paper is on the fragments of first-order logic complying with such conditions imposed on OBDA. Another focus is on the following problem: given theories $\mathcal{T}_1$, $\mathcal{T}_2$ and a signature Σ, decide whether, for all $\mathcal{A}$ and $\varphi(\boldsymbol{x})$ formulated in the language of Σ, we have $(\mathcal{T}_1, \mathcal{A}) \models \varphi(\boldsymbol{a})$ iff $(\mathcal{T}_2, \mathcal{A}) \models \varphi(\boldsymbol{a})$ (that is: $\mathcal{T}_1$ and $\mathcal{T}_2$ give the same answers to any Σ-query over the same Σ-data). In the *OWL 2* setting, an efficient solution to this problem would facilitate various stages of ontology developing and maintenance such as ontology versioning, refining, reusing, or module extraction [3].

References

1. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39, 385–429 (2007)
2. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: KR (2010)

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 6–7, 2011.

3. Kontchakov, R., Wolter, F., Zakharyaschev, M.: Logic-based ontology comparison and module extraction, with an application to DL-Lite. AI 174, 1093–1141 (2010)
4. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: IJCAI, pp. 2070–2075 (2009)

A Complexity Question in Justification Logic

Antonis Achilleos

The Graduate Center, CUNY
aachilleos@gc.cuny.edu

Abstract. Bounds for the computational complexity of major justification logics were found in papers by Buss, N. Krupski, Kuznets, and Milnikel: logics J, J4, JT, LP and JD, were established to be Σ_2^p-complete. A corresponding lower bound is also known for JD4, the system that includes the consistency axiom and positive introspection. However, no upper bound has been established so far for this logic. Here, the missing upper bound for the complexity of JD4 is established through an alternating algorithm. It is shown that using Fitting models of only two worlds is adequate to describe JD4; this helps to produce an effective tableau procedure and essentially is what distinguishes the new algorithm from existing ones.

Keywords: Justification Logic, Computational Complexity, Satisfiability.

1 Introduction

The classical analysis of knowledge includes the notion of justification, e.g., the famous tripartate view of knowledge as justified, true belief, usually attributed to Plato. Hintikka's modal logic approach represents knowledge as true belief. Justification logic extends epistemic logic by supplying the missing third component of Plato's characterization.

The Logic of Proofs LP was the first justification logic to be introduced, by Artemov, in [1,2] (see also [3]). Later, variations appeared in [5] corresponding to well-known normal modal logics. Several types of semantics are known for justification logics, but two are of interest in this paper: M-models, introduced by Mkrtychev ([12,17]) and F-models, introduced by Fitting ([4,8,9,14,18]). F-models resemble Kripke models for normal modal logics equipped with an additional mechanism, the admissible evidence function, usually denoted $\mathcal{A}$. For a term t and formula ϕ, $\mathcal{A}(t, \phi)$ will be the set of worlds in the model where t is appropriate evidence for ϕ. $t : \phi$ is true in a world iff ϕ is true in every accessible world and the world in question is in $\mathcal{A}(t, \phi)$. M-models are essentially F-models of only one world. In this setting $\mathcal{A}(t, \phi)$ will be either *true* or *false*.

Upper and lower bounds are known for the computational complexity of justification logics J, J4, JT, LP and JD, and determine the derivability problem for these to be Σ_2^p - complete ([6,10,12,13,15,16]). In [15], Kuznets presents a new algorithm for checking JD-satisfiability. This algorithm is in many perspectives similar to the

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 8–19, 2011.

ones for other justification logics mentioned here: J, J4, JT, LP. A tableau method is used to try and non-deterministically construct a model that satisfies the formula in question, then the algorithm checks whether the produced conditions for the admissible evidence function are legitimate, thus making the product of the tableau procedure, indeed, a model. This last check is known to be in NP ([10,13]) and is, in fact, NP-complete ([7]). Therefore, the resulting overall algorithm is a polynomial time alternating algorithm with one alternation, starting from a universal state and eventually reaching an existential state[1], which establishes that the problem is in Π_2^p and therefore the logic is in Σ_2^p.

The difference among the cases that have been dealt with previously lies in the consistent evidence property of logics JD and JD4 (the "D"); in an M-model of these logics, we can never have $\mathcal{A}(t, \perp)$. In other words, if $t_1 : \phi_1, \ldots, t_n : \phi_n$ are satisfied in a model, the set $\{\phi_1, \ldots, \phi_n\}$ must be consistent. To incorporate this condition, the algorithm continues and tries to verify whether this set, $\{\phi_1, \ldots, \phi_n\}$, is satisfiable in another model with another tableau construction. This is done by utilizing additional numerical prefixes on the prefixed formulas. A sequence of models is thus produced.

Although this construction, like all previous ones, is based on the compact character of M-models for justification logics, what it produces resembles something very similar to an F-model. In general, when discussing complexity issues for justification logics, working with F-models appears inconvenient and unnecessary: many possible worlds could succeed a current world. Furthermore, the admissible evidence function is defined on this multitude of worlds, which does not help to discuss complexity issues, especially when one is trying to confine the problem inside the Polynomial Hierarchy. On the other hand, M-models consist of only one world. The admissible evidence function is by far less complicated and the conditions it should satisfy can be checked by an NP-algorithm, except in the case of JD and JD4. For these logics, the additional condition that for any justification term t, $\mathcal{A}(t, \perp) = \mathit{false}$ cannot be verified as easily due to the negative nature of this condition. It would be nice to be able to sacrifice some, but not much, of the compact description of M-models for more convenient conditions on $\mathcal{A}$ and indeed, it seems that in the case of JD, this is exactly what can be done to provide a solution. Now, this idea will be taken a small step forward to provide a similar Σ_2 algorithm for JD4-satisfiability. Additionally, the positive introspection axiom of JD4 will help provide an even simpler class of models than in the case of JD, making the study of its complexity easier. Specifically, it is discovered that using Fitting-like models of only two worlds is adequate to describe JD4.

2 The Logic JD4

JD4 was first introduced in [5] as a variation of LP, the Logic of Proofs. It is the explicit counterpart of D4, both in intuition, as there is some similarity between their axioms, and in a more precise way (see [5]).

[1] Or, this can be viewed as a coNP algorithm using an oracle from NP.

The language will include justification constants $c_i,\ i \in \mathbb{N}$, justification variables: $x_i,\ i \in \mathbb{N}$ and justification terms, usually denoted $t, s, \ldots$. These are defined as follows.

Definition 1 (Justification Terms)

- *Constants ($c_i,\ i \in \mathbb{N}$) and variables ($x_i,\ i \in \mathbb{N}$) are terms;*
- *If t_1, t_2 are terms, so are*

$$(t_1 \cdot t_2),\ (t_1 + t_2),\ (!t_1).$$

"$\cdot$" is usually called application, "+" is called sum, and "!" proof checker. The set of justification terms will be called Tm.

Also, propositional variables will be used in the language: $p_i,\ i \in \mathbb{N}$. The set of propositional variables will be called $SLet$. The formulas of the language are

Definition 2 (Justification Formulas). *The formulas of the language are defined recursively:*
If p is a propositional variable, t is a term and ϕ, ψ are formulas, then so are

$$p,\ \bot,\ (\phi \rightarrow \psi),\ (t{:}\phi)$$

$\neg\phi$ can be seen as short for $\phi \rightarrow \bot$, and the rest of the connectives can be defined from these in the usual way. Also as usual, parentheses will be omitted using standard conventions, and naturally, $!s{:}s{:}\phi$ will be read as $(!s{:}(s{:}\phi))$. Fm will denote the set of justification formulas.

The axioms of $\mathsf{JD4}_\emptyset$ are the following.

A1 Finitely many schemes of classical propositional logic;
A2 $s{:}(\phi \rightarrow \psi) \rightarrow (t{:}\phi \rightarrow (s \cdot t){:}\psi)$ - Application Axiom;
A3

$$\begin{aligned} &s{:}\phi \rightarrow (s+t){:}\phi \\ &s{:}\phi \rightarrow (t+s){:}\phi \qquad \text{- Monotonicity Axiom;} \end{aligned}$$

A5 $t{:}\phi \rightarrow !t{:}t{:}\phi$ - Positive introspection;
A6 $t{:}\bot \rightarrow \bot$ - Consistency Axiom
and Modus Ponens.
MP Modus Ponens Rule :

$$\frac{\phi \rightarrow \psi \qquad \phi}{\psi}.$$

Definition 3. *A constant specification for a justification logic JL is any set*

$$\mathcal{CS} \subseteq \{c{:}A \mid c \text{ is a constant, } A \text{ an axiom of } JL\}.$$

A c.s. is axiomatically appropriate *if each axiom is justified by at least one constant,* schematic *if every constant justifies a certain number of axiom schemes (0 or more), and* schematically injective *if it is schematic and every constant justifies at most one scheme.*

Definition 4. *Given a constant specification* $\mathcal{CS}$ *for* $\mathsf{JD4}$*, the logic* $\mathsf{JD4}_{\mathcal{CS}}$ *is* $\mathsf{JD4}_{\emptyset}$*, with the additional rule*

$$\frac{}{c\colon A}\ \mathrm{R4}_{\mathcal{CS}}$$

where $c\colon A \in \mathcal{CS}$*.*

A definition of the (Fitting) semantics for $\mathsf{JD4}_{\mathcal{CS}}$ follows.

Definition 5 ([4,8,9,14,18]). *An F-model* $\mathcal{M}$ *for* $\mathsf{JD4}_{\mathcal{CS}}$ *is a quadruple* $(W, R, V, \mathcal{A})$*, where* $W \neq \emptyset$ *is the set of worlds (or states) of the model,* R *is a transitive and serial (for any* $a \in W$ *there is some* $b \in W$ *such that* aRb*) binary relation on* W*,* V *assigns a subset of* W *to each propositional variable,* p*, and* $\mathcal{A}$*, the admissible evidence function, assigns a subset of* W *to each pair of a justification term and a formula. Additionally,* $\mathcal{A}$ *must satisfy the following conditions:*

Application closure: *for any formulas* ϕ, ψ *and justification terms* t, s*,*

$$\mathcal{A}(s, \phi \to \psi) \cap \mathcal{A}(t, \phi) \subseteq \mathcal{A}(s \cdot t, \psi).$$

Sum closure: *for any formula* ϕ *and justification terms* t, s*,*

$$\mathcal{A}(t, \phi) \cup \mathcal{A}(s, \phi) \subseteq \mathcal{A}(t + s, \phi).$$

Simplified $\mathcal{CS}$-closure: *for any axiom* A*, constant* c*, such that* $c\colon A \in \mathcal{CS}$*,*

$$\mathcal{A}(c, A) = W.$$

Positive introspection closure: *for any formula* ϕ *and justification term* t*,*

$$\mathcal{A}(t, \phi) \subseteq \mathcal{A}(!t, t\colon\phi).$$

Monotonicity: *for any formula* ϕ*, justification term* t *and* $a, b \in W$*, if* aRb *and* $a \in \mathcal{A}(t, \phi)$*, then* $b \in \mathcal{A}(t, \phi)$*.*

Truth in the model is defined in the following way, given a state a*:*

- $M, a \not\models \bot$.
- *If* p *is a propositional variable, then* $\mathcal{M}, a \models p$ *iff* $a \in V(p)$
- *If* ϕ, ψ *are formulas, then* $\mathcal{M}, a \models \phi \to \psi$ *if and only if* $M, a \models \psi$*, or* $\mathcal{M}, a \not\models \phi$*.*
- *If* ϕ *is a formula and* t *a term, then* $\mathcal{M}, a \models t\colon\phi$ *if and only if* $a \in \mathcal{A}(t, \phi)$ *and for all* $b \in W$*, if* aRb*, then* $\mathcal{M}, b \models \phi$*.*

Proposition 1 ([14]). $\mathsf{JD4}_{\mathcal{CS}}$ *is sound and complete w.r.t. its F-models, for an axiomatically appropriate constant specification. Additionally, it is complete w.r.t. its F-models that satisfy the following property.*

Strong Evidence Property: $\mathcal{M}, a \models t\colon\phi$ *if and only if* $a \in \mathcal{A}(t, \phi)$*.*

It is also useful to present Mkrtychev (M-) models for $\mathsf{JD4}$. M-models are F-models with just one world. However, since if we insisted on seriality, we would introduce factivity ($t : \phi \rightarrow \phi$ would be valid), this condition is replaced by another, the *consistent evidence condition* (in the definition below).

Definition 6 ([12,17]). *An* M-model for $\mathsf{JD4}_{\mathcal{CS}}$*, where* $\mathcal{CS}$ *is a constant specification for* $\mathsf{JD4}$ *is a pair*

$$\mathcal{M} = (V, \mathcal{A}),$$

where propositional valuation

$$V : SLet \longrightarrow \{true, false\}$$

assigns a truth value to each propositional variable and

$$\mathcal{A} : Tm \times Fm \longrightarrow \{true, false\}$$

is an admissible evidence function*.* $\mathcal{A}(t, \phi)$ *will be used as an abbreviation for* $\mathcal{A}(t, \phi) = true$ *and* $\neg\mathcal{A}(t, \phi)$ *as an abbreviation for* $\mathcal{A}(t, \phi) = false$*.*
The admissible evidence function must satisfy certain closure conditions:

Application Closure: *If* $\mathcal{A}(s, \phi \rightarrow \psi)$ *and* $\mathcal{A}(t, \phi)$ *then* $\mathcal{A}(s \cdot t, \psi)$*.*
Sum Closure: *If* $\mathcal{A}(s, \phi)$ *then* $\mathcal{A}(s + t, \phi)$*.*
If $\mathcal{A}(t, \phi)$ *then* $\mathcal{A}(s + t, \phi)$*.*
Simplified $\mathcal{CS}$ **Closure:** *If* $c : A \in \mathcal{CS}$*, then* $\mathcal{A}(c, A)$*.*
Positive Introspection Closure: *If* $\mathcal{A}(t, \phi)$ *then* $\mathcal{A}(!t, t : \phi)$
Consistent Evidence Condition: $\mathcal{A}(t, \bot) = false$*,*

for any formulas ϕ, ψ*, any terms* s, t *and any* $c : A \in \mathcal{CS}$*.*
The truth relation $\mathcal{M} \models H$ *is defined as follows:*

- $\mathcal{M} \models p$ *iff* $V(p) = true$
- $\mathcal{M} \not\models \bot$
- $\mathcal{M} \models \phi \rightarrow \psi$ *iff* $\mathcal{M} \not\models \phi$ *or* $\mathcal{M} \models \psi$
- $\mathcal{M} \models t : \phi$ *iff* $\mathcal{A}(t, \phi)$

for any formulas ϕ, ψ*, any term* t *and any propositional variable* p*.*

Proposition 2 ([12]). $\mathsf{JD4}_{\mathcal{CS}}$ *is sound and complete with respect to its M-models.*

Note 1. The *Simplified* $\mathcal{CS}$*-closure* condition is called *simplified*, because it replaces another, less simple, condition called simply $\mathcal{CS}$*-closure*. $\mathcal{CS}$*-closure* is necessary when dealing with justification logics without positive introspection, whose models do not have the *Positive Introspection Closure* condition. Since *Simplified* $\mathcal{CS}$*-closure* is indeed simpler than $\mathcal{CS}$*-closure* and under the presence of *Positive Introspection Closure* can produce it, we prefer this simplified version.

The following definition will prove useful later on.

Definition 7. *Given an F-model* $\mathcal{M}$*, a world* w *of the model and a formula* $t : \phi$*, we say that* $t : \phi$ *is* factive *at world* w *of* $\mathcal{M}$*, if and only if* $\mathcal{M}, w \models t : \phi \rightarrow \phi$*. Similarly, we can define when* $t : \phi$ *is factive in an M-model. A set* Φ *of formulas of the form* $t : \phi$ *will be factive exactly when all elements of* Φ *are factive.*

3 $*$-Calculus and Minimal Evidence Functions

In this section, the $*$-calculus is defined. The $*$-calculus provides an independent axiomatization of the reflected fragments of justification logics and is an invaluable tool in the study of the complexity of these logics. The concepts, notation and results in this section come from [11,13,17].

Definition 8 (Star-expressions). *If t is a term and ϕ is a formula, then $*(t,\phi)$ is a star-expression ($*$-expression).*

Definition 9. *For any justification logic L and constant specification $\mathcal{CS}$, the reflected fragment of $L_{\mathcal{CS}}$ is*

$$rL_{\mathcal{CS}} = \{t{:}\phi \mid L_{\mathcal{CS}} \vdash t{:}\phi\}.$$

Definition 10 ($*$-calculus)

$*\mathcal{CS}$ *Axioms: $*(c,A)$, where $c{:}A \in \mathcal{CS}$.*
$*A2$

$$\frac{*(s,\phi \to \psi) \qquad *(t,\phi)}{*(s\cdot t,\psi)}$$

$*A3$

$$\frac{*(t,\phi)}{*(s+t,\phi)} \qquad \frac{*(s,\phi)}{*(s+t,\phi)}$$

$*A4$

$$\frac{*(t,\phi)}{*(!t,t{:}\phi)}$$

The calculus: *The $*!_{\mathcal{CS}}$-calculus is a calculus on starred expressions and includes $*\mathcal{CS}, *A2, *A3$ and $*A4$.*

Theorem 1 ([10,13]). *For any constant specification $\mathcal{CS}$,*

$$\mathsf{rJD4}_{\mathcal{CS}} \vdash t{:}\phi \iff \mathsf{rJD4}_{\mathcal{CS}} \vdash t{:}\phi \iff \vdash_{*!_{\mathcal{CS}}} *(t,\phi).$$

Possible evidence functions are presented next, together with a way to produce a minimal evidence function for an M-model.

Definition 11. *An M-type possible evidence function is any function*

$$\mathcal{B} : Tm \times Fm \longrightarrow \{\mathit{true}, \mathit{false}\}.$$

A possible evidence function is essentially an admissible evidence function with no conditions imposed on it.

Definition 12. *We say that an M-type possible evidence function $\mathcal{B}_2$ is based on an M-type possible evidence function $\mathcal{B}_1$ and write*

$$\mathcal{B}_1 \subseteq \mathcal{B}_2,$$

if for all terms t and formulas ϕ,

$$\mathcal{B}_1(t,\phi) \Longrightarrow \mathcal{B}_2(t,\phi).$$

Definition 13. *Let $\mathcal{EF}$ be a class of M-type possible evidence functions. A possible evidence function $\mathcal{B} \in \mathcal{EF}$ is called the minimal evidence function in $\mathcal{EF}$ if for all $\mathcal{B}' \in \mathcal{EF}$,*

$$\mathcal{B} \subseteq \mathcal{B}'.$$

Definition 14. *Given a possible evidence function $\mathcal{B}$, let $\mathcal{B}^* = \{*(t,\phi) | \mathcal{B}(t,\phi) = true\}$.*

Theorem 2 ([17,13]). *For any constant specification $\mathcal{CS}$ for* JD4 *and any possible evidence function $\mathcal{B}$, if the class of M-type admissible evidence functions for* $\mathsf{JD4}_{\mathcal{CS}}$ *based on $\mathcal{B}$ is nonempty, then it has a unique minimal element $\mathcal{A}$, which is the following:*

$$\mathcal{A}(t,\phi) \Longleftrightarrow \mathcal{B}^* \vdash_{*!_{\mathcal{CS}}} *(t,\phi).$$

Note 2. In the following, an F-type admissible evidence function when considering a single world of the model may be treated as an M-type admissible evidence function, when the appropriate conditions are met. Despite changes in notation, under certain circumstances this is entirely acceptable and in fact this change in perspective will be very useful and frequent. Finally, it is useful, given an F-type admissible evidence function $\mathcal{A}$ and a world u, to define $\mathcal{A}_u$ to be the set $\{(t,\phi) | u \in \mathcal{A}(t,\phi)\}$.

Finally, since we are discussing complexity issues, it is natural that the following theorem is relevant. In fact, it will prove to be extremely useful later on.

Theorem 3 ([10,13]). *Let $\mathcal{CS}$ be a schematic constant specification decidable in polynomial time. Then, there exists a non-deterministic algorithm that runs in polynomial time and determines, given a finite set S of $*$-expressions, a formula ϕ and a term t, whether*

$$S \vdash_{*!_{\mathcal{CS}}} *(t,\phi).$$

4 A Class of Models

In the following, the constant specification $\mathcal{CS}$ will be assumed to be axiomatically appropriate and, when discussing complexity issues, efficiently decidable. The algorithm that will be presented and its correctness will be based on the following proposition.

Proposition 3. *A formula ϕ is $\mathsf{JD4}_{\mathcal{CS}}$-satisfiable if and only if it is satisfiable by an F-model $\mathcal{M} = (W, R, V, \mathcal{A})$ for $\mathsf{JD4}_{\mathcal{CS}}$ that additionally has the following properties:*

- *W has exactly two elements, a, b.*
- *$R = \{(a, b), (b, b)\}$.*

Proof. Let ϕ be a formula that is $\mathsf{JD4}_{\mathcal{CS}}$-satisfiable and let $\mathcal{M}^* = (W, R, V, \mathcal{A})$ be a model and $a \in W$ a world of the model that satisfies ϕ. Assume that $\mathcal{M}^*$ satisfies the Strong Evidence Property.

We know that R is serial and transitive and that $\mathcal{A}$ satisfies the monotonicity property. From this, we know that there is an infinite sequence of elements of W, $\alpha = (a_i)_{i \in \mathbb{N}}$, such that $a_0 = a$, $i < j \Rightarrow a_i R a_j$ & $\mathcal{A}_{a_i} \subseteq \mathcal{A}_{a_j}$.

For any $t : F$, there is at most one $j \in \mathbb{N}$, $\mathcal{M}^*, a_j \not\models t : F \rightarrow F$. Otherwise, there are $i < j$ s.t. $\mathcal{M}^*, a_i, a_j \not\models t : F \rightarrow F$. Since $\mathcal{M}^*, a_i \not\models t : F \rightarrow F$, we have $\mathcal{M}^*, a_i \models t : F$. From this, it follows that $\mathcal{M}^*, a_j \models F$, so $\mathcal{M}^*, a_j \models t : F \rightarrow F$ - a contradiction.

Therefore, for any finite set of term-prefixed formulas, there is an i, after which for all terms c of sequence α that set is factive at c. More specifically, let Φ be the set of term-prefixed subformulas of ϕ and let b be a term of sequence α, where Φ is factive.

Define $\mathcal{M}$ to be the model $(\{a, b\}, \{(a, b), (b, b)\}, V', \mathcal{A}')$, such that $V', \mathcal{A}'$ agree with $V, \mathcal{A}$ on a, b. That is, for any $w \in \{a, b\}$, t term, ψ formula, p propositional variable, $w \in V(p)$ if and only if $w \in V'(p)$, and $w \in \mathcal{A}(t, \psi)$ if and only if $w \in \mathcal{A}'(t, \psi)$. It is easy to see that the new model satisfies the conditions required of F-models for $\mathsf{JD4}_{\mathcal{CS}}$[2].

By induction on the structure of χ, we can show that for any χ, subformula of ϕ, $\mathcal{M}^*, b \models \chi$ iff $\mathcal{M}, b \models \chi$ (and the propositional cases are trivial, so). If $\chi = t : \omega$, then $\mathcal{M}^*, b \models \chi$ iff $\mathcal{M}^*, b \models t : \omega$ iff $\mathcal{M}^*, b \models \omega$ and $b \in \mathcal{A}(t, \omega)$ (Strong Evidence) iff $\mathcal{M}, b \models \omega$ and $b \in \mathcal{A}'(t, \omega)$ iff $\mathcal{M}, b \models \chi$.

To prove that $\mathcal{M}, a \models \phi$, we will first prove that

$$\mathcal{M}^*, a \models \psi \Leftrightarrow \mathcal{M}, a \models \psi,$$

for any ψ subformula of ϕ, by induction on the structure of ψ. If ψ is a propositional variable, and for the propositional cases, again, this is obvious and the only interesting case is when $\psi = t : \chi$. In this case, $\mathcal{M}^*, a \models t : \chi$ iff $a \in \mathcal{A}(t, \chi)$ and $M^*, b \models \chi$ iff $a \in \mathcal{A}'(t, \chi)$ and $M, b \models \chi$ iff $\mathcal{M}, a \models t : \chi$. □

Observation 1. Note that we can now replace the admissible evidence function with another, say $\mathcal{A}^m$, such that $w \in \mathcal{A}^m(t, \psi)$ iff $\mathcal{M}, w \models t : \psi$. This new function will satisfy the necessary conditions to be an admissible evidence function and the new model will satisfy the same formulas as the old one in the same worlds. Therefore, we can claim the following corollary.

[2] Of course, we assume here that $a \neq b$, but this is a legitimate assumption. If we need to make this explicit, we could simply have $W = \{(a, 0), (b, 1)\}$ and the accessibility relation, V', A' behave in the same way.

Corollary 1. *A formula is* $\mathsf{JD4}_{\mathcal{CS}}$*-satisfiable if and only if it is satisfiable by an F-model* $\mathcal{M} = (W, R, V, \mathcal{A})$ *for* $\mathsf{JD4}_{\mathcal{CS}}$ *that additionally has the following properties:*

– *W has exactly two elements, a, b;*
– *$R = \{(a, b), (b, b)\}$;*
– *$a \in \mathcal{A}(t, F)$ if and only if $\mathcal{M}, a \models t : F$ for all $t : F$. (Strong Evidence Condition)*

5 The Algorithm and Its Analysis

Neither the algorithm that determines $\mathsf{JD4}_{\mathcal{CS}}$-satisfiability nor its analysis is particularly novel (c.f. [12,13,15]). It is in fact based on the ones already used to establish the same upper bound for the satisfiability for J, J4, JT, LP, except for certain differences that stem from the fact that we are discussing a different logic and thus the algorithm is based on a different class of models. It is based on a tableau construction.

Prefixed expressions will be used and there will be two types of prefixes and two types of expressions. The first will be the usual T or F prefix and the other will be the prefix that will intuitively denote the world we are referring to; these are a and b. So, the prefixed formulas will be of the form $w\ P\ e$, where $w \in \{a, b\}, P \in \{T, F\}$ and e is either a formula of the language or a $*$-expression. w will be called the world prefix and P the truth prefix.

As was mentioned previously, the algorithm will be based on a tableau construction. The propositional tableau rules will be the ones usually used and they are not mentioned here. The non-propositional cases are covered by the following rules.

$$\frac{a\ T\ s{:}\psi}{\begin{array}{c} a\ T\ \ *(s,\psi) \\ b\ T\ s{:}\psi \end{array}}, \qquad \frac{b\ T\ s{:}\psi}{\begin{array}{c} b\ T\ \ *(s,\psi) \\ b\ T\ \psi \end{array}},$$

$$\frac{a\ F\ s{:}\psi}{a\ F\ \ *(s,\psi) \mid b\ F\ \psi}, \qquad \frac{b\ F\ s{:}\psi}{b\ F\ \ *(s,\psi) \mid b\ F\ \psi}$$

The algorithm runs in two phases.

If ϕ is the formula which must be checked for $\mathsf{JD4}_{\mathcal{CS}}$-satisfiability, then during the first phase, the algorithm will construct a tableau branch, starting from just $T\ \phi$ using the tableau rules to produce more prefixed formulas in a non-deterministic way; the vertical bars appearing at the rules denote a non-deterministic choice. After all possible tableau derivations have been applied, there are two possibilities for the constructed branch. It can either be propositionally closed, that is, it could contain $w\ T\ e$ and $w\ F\ e$, or it can be complete, that is, the branch is not propositionally closed and no application of a tableau rule gives a new prefixed formula. If it is propositionally closed, the input is rejected, otherwise, the second phase of the algorithm begins. Let X_a be the set of star expressions prefixed with $a\ T$ and X_b the set of star expressions prefixed with $b\ T$ in the branch. Confirm that no expression of the form $*(t, \phi)$ that appears negatively in the branch for world prefix w can be derived from X_w. If this is indeed the case, the algorithm accepts, otherwise, it rejects.

The proof of the correctness of the algorithm follows.

Proof. Supposing formula ϕ is satisfiable by $\mathcal{M} = (\{a, b\}, \{(a, b), (b, b)\}, V, \mathcal{A})$ such as the ones described above and starting the procedure with $a\ T\ \phi$, it is easy to see that there is a way to perform the tableau rules while producing a-prefixed expressions satisfied at world a and b-prefixed expressions satisfied at world b. Now, if X_w derives $*(t, \psi)$, then the minimal evidence function that includes X_w should also include (t, ψ) (by Theorem 2), and therefore so should $\mathcal{A}$ in the corresponding world. Therefore, no such negative expression will be derivable by X_w. In conclusion, the algorithm accepts.

On the other hand, suppose the algorithm accepts. Suppose a complete branch of the tableau that is produced in an accepting branch of the computation tree. A model will be constructed to satisfy ϕ. This will be $\mathcal{M} = (\{a, b\}, \{(a, b), (b, b)\}, V, \mathcal{A})$. $V(p)$ will include a iff $a\ T\ p$ appears in the tableau and similarly for b. $\mathcal{A}$ on a will be the minimal evidence function based on the possible evidence function which maps (t, ψ) to *true* iff $a\ T\ *(t, \psi)$ appears in the tableau and again, similarly for b.[3] By the tableau rules, $X_a \subseteq X_b$, because whenever a formula $a\ T\ *(s, \psi)$, is produced, so is $b\ T\ *(s, \psi)$ and therefore monotonicity is satisfied. If $\mathcal{A}$ so defined includes any negative $*$-expression in any of the two worlds, the second step of the algorithm would have rejected the input and the computation branch would not be accepting (again, by Theorem 2).

The model satisfies at a all a-prefixed expressions and at world b all b-prefixed expressions. This can be proven by induction on the structure of the expressions. By the above argument, this is automatically true for starred expressions. Also, by definition of V, this is true for propositional variables. Propositional cases are easy, so it remains to show this for formulas of the form $t{:}\psi$.

First, for b-prefixed formulas. If $b\ T\ t{:}\psi$ is in the branch, there must also be $b\ T\ *(t, \psi)$ and $b\ T\ \psi$. By I.H., these are already satisfied, so $b \in \mathcal{A}(t, \psi)$ and $\mathcal{M}, b \models \psi$. Therefore, $\mathcal{M}, b \models t{:}\psi$. If $b\ F\ t{:}\psi$ is in the branch, then the branch must also include either $b\ F\ \psi$ or $b\ F\ *(t, \psi)$. In either case, the conclusion is $\mathcal{M}, b \not\models t{:}\psi$.

[3] Technically, we do not know at this point whether there actually exists any admissible evidence function based on that possible evidence function - call it B. The problem is that perhaps $B^* \vdash_{\mathcal{CS}} *(t, \perp)$ and therefore $\mathcal{A}$ on a should be defined as the possible evidence function mapping (t, ψ) to *true* iff $B^* \vdash_{\mathcal{CS}} *(t, \psi)$. However, as it turns out, $\mathcal{M}$ is a JD4 model and from this proof if $B(t, \psi) = \mathit{true}$, then $a\ T\ *(t, \psi)$ appears in the tableau and consequently $\mathcal{M}, a \models t{:}\psi$. So, $\mathcal{A}'(t, \psi) = \mathit{true}$ iff $\mathcal{M}, a \models t{:}\psi$ is an admissible evidence function based on B. This means that, in retrospective, $\mathcal{A}$ on a as defined above is an admissible evidence function, thus justifying treating it as such.

Finally, the case of a-prefixed formulas. If $a\ T\ t{:}\psi$ is in the branch, there must also be $a\ T\ *(t, \psi)$ and $b\ T\ \psi$ (and $b\ T\ t{:}\psi$ too, but it is not relevant here). By I.H., these are already satisfied, so $a \in \mathcal{A}(t, \psi)$ and $\mathcal{M}, b \models \psi$. Therefore, $\mathcal{M}, a \models t:\psi$. If $a\ F\ t:\psi$ is in the branch, then the branch must also include either $b\ F\ \psi$ or $a\ F\ *(t, \psi)$. In either case, the conclusion is $\mathcal{M}, b \not\models t{:}\psi$.

This completes the correctness proof of the algorithm. □

The first phase of the algorithm runs in nondeterministic polynomial time, while the second checks a condition known to be in coNP (Theorem 3). Therefore, the algorithm establishes that JD4-satisfiability is in Σ_2^p. The following corollary follows immediately.

Corollary 2. $\mathsf{JD4}_{\mathcal{CS}}$ *is in* Π_2^p *for any axiomatically appropriate, schematic, and efficiently decidable* $\mathcal{CS}$.

The following has been recently proven in [6].

Theorem 4 ([6]). $\mathsf{JD4}_{\mathcal{CS}}$ *is* Π_2^p*-hard, for any axiomatically appropriate, and schematically injective* $\mathcal{CS}$.

Finally, combining these two results, we can claim the following.

Corollary 3. $\mathsf{JD4}_{\mathcal{CS}}$ *is* Π_2^p*-complete, for any axiomatically appropriate, schematically injective, and efficiently decidable* $\mathcal{CS}$.

Acknowledgements. The author would like to thank Sergei Artemov for his encouragement and essential suggestions and Karen Kletter for her invaluable help in editing.

References

1. Artemov, S.: Operational modal logic. Technical Report MSI 95–29, Cornell University (1995)
2. Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
3. Artemov, S.: Kolmogorov and Gödel's approach to intuitionistic logic: current developments. Russian Mathematical Surveys 59(2), 203–229 (2004)
4. Artemov, S.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
5. Brezhnev, V.N.: On explicit counterparts of modal logics. Technical Report CFIS 2000–05, Cornell University (2000)
6. Buss, S.R., Kuznets, R.: Lower complexity bounds in justification logic (2009) (manuscript)
7. Buss, S.R., Kuznets, R.: The NP-completeness of reflected fragments of justification logics. In: Artëmov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 122–136. Springer, Heidelberg (2008)
8. Fitting, M.: A semantics for the Logic of Proofs. Technical Report TR–2003012, CUNY Ph.D. Program in Computer Science (2003)

9. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
10. Krupski, N.V.: On the complexity of the reflected logic of proofs. Theoretical Computer Science 357(1-3), 136–142 (2006)
11. Krupski, V.N.: Referential logic of proofs. Theoretical Computer Science 357(1-3), 143–166 (2006)
12. Kuznets, R.: On the complexity of explicit modal logics. In: Proceedings of the 14th Annual Conference of the EACSL on Computer Science Logic, pp. 371–383. Springer, Heidelberg (2000)
13. Kuznets, R.: Complexity Issues in Justification Logic. PhD thesis, CUNY Graduate Center (2008)
14. Kuznets, R.: Self-referentiality of justified knowledge. In: Hirsch, E.A., Razborov, A.A., Semenov, A., Slissenko, A. (eds.) CSR 2008. LNCS, vol. 5010, pp. 228–239. Springer, Heidelberg (2008)
15. Kuznets, R.: Complexity through tableaux in justification logic. In: 2008 European Summer Meeting of the ASL, Logic Colloquium (2008); Bulletin of Symbolic Logic 15(1), 121 (2009)
16. Milnikel, R.S.: Derivability in certain subsystems of the logic of proofs is Π_2^p-complete. Annals of Pure and Applied Logic 145(3), 223–239 (2007)
17. Mkrtychev, A.: Models for the logic of proofs. In: Adian, S., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)
18. Pacuit, E.: A note on some explicit modal logics. In: Proceedings of the 5th Panhellenic Logic Symposium (2005)

Basic Model Theory for Memory Logics

Carlos Areces[1], Facundo Carreiro[2,⋆], Santiago Figueira[2,3,⋆⋆],
and Sergio Mera[2,⋆]

[1] INRIA Nancy Grand Est, Nancy, France
areces@loria.fr
[2] Dto. Computación, FCEN, Universidad de Buenos Aires, Argentina
[3] CONICET, Argentina
{fcarreiro,santiago,smera}@dc.uba.ar

Abstract. Memory logics is a family of modal logics whose semantics is specified in terms of relational models enriched with additional *data structure* to represent a memory. The logical language includes a collection of operations to access and modify the data structure. In this paper we study basic model properties of memory logics, and prove results concerning characterization, definability and interpolation. While the first two properties hold for all memory logics introduced in this article, interpolation fails in most cases.

1 Introduction

In the last decades, *modal logics* have become a wide collection of formal systems with some general aspects in common: they are usually interpreted over relational structures, they are generally computationally well behaved, and they take a local perspective when evaluating a formula. Nowadays, the practical influence of modal logics is undeniable as they are used in many applications like linguistics, artificial intelligence, knowledge representation, specification and software verification, etc. (see [6] for details).

In a number of recent papers [4,3,2,1] we have investigated a family of logics called *memory logics*, extending the classical modal logic.[1] Intuitively, memory logics enrich the standard relational models used by most modal logics with a *data structure*. The logical language is then extended with a collection of operations to access and modify this data structure. In this article we fix the data structure to be a *set*, but other structures are analyzed in [1].

Assume as given a signature $\mathcal{S} = \langle \text{PROP}, \text{REL} \rangle$ that defines the sets of propositional and relational symbols, respectively. Let $\mathcal{N}$ be a standard relational model over $\mathcal{S}$, i.e., $\mathcal{N} = \langle \mathcal{F}, V \rangle$, where $\mathcal{F} = \langle W, (R_r)_{r \in \text{REL}} \rangle$ is a suitable frame (i.e. W is a nonempty set whose elements we will call *states*, and $R_r \subseteq W^2$ for each

⋆ F. Carreiro and S. Mera were partially supported by CONICET Argentina.
⋆⋆ S. Figueira was partially supported by CONICET Argentina (grant PIP 370).

[1] Due to lack of space we will assume in this article that the reader is familiar with modal logics, see [6,5] for complete details.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 20–34, 2011.

$r \in \textsc{rel}$, which we call *accessibility relations*) and $V : \textsc{prop} \to 2^W$ is the valuation function. We obtain a model for memory logics, extending this structure with a set $S \subseteq W$ representing the current 'memory' of the model. For $\mathcal{M}$ an arbitrary model, we will denote its domain by $|\mathcal{M}|$, and we will usually represent a model $\langle\langle W, R\rangle, V, S\rangle$ simply as $\langle W, R, V, S\rangle$.

A set is a very simple data structure (e.g., compare it with a list, a tree, etc), but even in this setting, we can define a set of operators that interacts with the memory in different ways. One can think of different types of simple updates that can be performed on the memory of a model: to store or delete an element, to clean the memory, etc. If $\mathcal{M} = \langle \mathcal{F}, V, S\rangle$ is a model for memory logics as defined above, we define

$$\mathcal{M}[*] = \langle \mathcal{F}, V, \emptyset\rangle; \quad \mathcal{M}[w] = \langle \mathcal{F}, V, S \cup \{w\}\rangle; \quad \mathcal{M}[-w] = \langle \mathcal{F}, V, S \setminus \{w\}\rangle.$$

Let $\mathcal{M}[w_1, \ldots, w_n]$ be a shorthand for $((\mathcal{M}[w_1]) \ldots)[w_n]$. Besides the standard Boolean and diamond operators of the basic modal logic, we define

$$\begin{array}{llll} \mathcal{M}, w \models \textcircled{r}\varphi & \text{iff } \mathcal{M}[w], w \models \varphi & \mathcal{M}, w \models \textcircled{f}\varphi & \text{iff } \mathcal{M}[-w], w \models \varphi \\ \mathcal{M}, w \models \textcircled{e}\varphi & \text{iff } \mathcal{M}[*], w \models \varphi & \mathcal{M}, w \models \textcircled{k} & \text{iff } w \in S \end{array}$$

The 'remember' operator $\textcircled{r}$ (a unary modality) marks the current state as being 'already visited', by storing it in S. In contrast, the 'forget' operator $\textcircled{f}$ removes the current state from the memory, while the 'erase' operator $\textcircled{e}$ wipes out the memory. These are the operators we use to update the memory. On the other hand, the zero-ary operator $\textcircled{k}$ (for 'known') queries S to check if the current state is in the memory.[2]

Besides these basic operators, we can also impose constraints on the interplay between memory storage and the standard modalities. There are some contexts when we do not need $\textcircled{r}$ and $\langle r\rangle$ as two separate operators: we are only interested in the trail of memorized points we used to evaluate a formula (for details on possible applications see [13]). In these cases the $\langle\!\langle r\rangle\!\rangle$ operator will be handy:

$$\mathcal{M}, w \models \langle\!\langle r\rangle\!\rangle\varphi \text{ iff } \exists w' \in W, wR_r w' \text{ and } \mathcal{M}[w], w' \models \varphi$$

That is, $\langle\!\langle r\rangle\!\rangle\varphi$ is equivalent to $\textcircled{r}\langle r\rangle\varphi$. We will denote the dual of this operator as $[\![r]\!]$, with the usual interpretation. As it was showed in [1], this operator is very useful to regain decidability for some fragments of memory logic.

A particularly interesting class of models to investigate is the class $\mathcal{C}_\emptyset$ where the memory is empty, i.e., $\mathcal{C}_\emptyset = \{\mathcal{M} \mid \mathcal{M} = \langle \mathcal{F}, V, \emptyset\rangle\}$. It is natural to consider starting to evaluate a formula in a model of $\mathcal{C}_\emptyset$, as it is over $\mathcal{C}_\emptyset$ that the operators $\textcircled{k}$ and $\textcircled{r}$ have the most natural interpretation. As it is shown in [1], the restriction to this class has important effects on expressivity and decidability. It is worth noting that a formula is *initially* evaluated in a model of $\mathcal{C}_\emptyset$, but during the evaluation the model can change to one with nonempty memory. This dynamic behavior is a distinctive feature of memory logics over the classical modal logic: the value of S changes as the evaluation of the formula proceeds. This is

[2] Notice that all these operators are self dual.

not different to what happens with an assignment during the evaluation of a first order formula.

It is well known that a classical modal model $\mathcal{M} = \langle \mathcal{F}, V \rangle$ can be seen as a first order model over an appropriate signature, and that there is a standard translation ST_x transforming every modal formula φ into a first order formula $\mathsf{ST}_x(\varphi)$ with x as its only free variable such that $\mathcal{M}, w \models \varphi$ iff $\mathcal{M}, g^x_w \models \mathsf{ST}_x(\varphi)$, where g^x_w is an arbitrary assignment that maps x to w (on the left, $\mathcal{M}$ should be considered as a first order model, and $\models$ as the standard first order satisfiability relation). Similarly, any memory model can be seen as a first order model, and we can define a translation which transforms memory formulas into equivalent first order formulas (for more details see [1]). We will use this result for some results in this article.

In [4,1] some computational aspects of memory logics were studied, together with results for separating different memory logics in terms of expressive power. In [2,1] the focus was put in proof theoretical results. In this article we analyze some important theorems of the basic model theory for memory logics. The main tool for all our results on characterization, definability and interpolation is the notion of *bisimulation*. In Section 2 we present suitable notions of bisimulation for different memory logics. In Section 3 we state a van Benthem like characterization theorem for memory logics and we study when a class of memory models is definable by a set of memory formulas, or by a single formula. In Section 4, we analyze the validity of the Craig interpolation theorem for many members of the family of memory logics. Finally, in Section 5 we discuss further work and draw some conclusions.

Notation. As we will be discussing many different logics, we introduce here some notational conventions. We call $\mathcal{ML}$ the basic modal logic, and add a superscript m to indicate the addition of a memory-set and the basic memory operators $\textcircled{\texttt{r}}$ and $\textcircled{\texttt{k}}$. Additional operators included in the language are listed explicitly. Since we can choose to use $\langle r \rangle$ or $\langle\!\langle r \rangle\!\rangle$, we will also include the diamond explicitly in this list. For example, $\mathcal{ML}^m(\langle r \rangle, \textcircled{\texttt{e}})$ is the modal logic with the standard diamond operator extended with $\textcircled{\texttt{r}}$, $\textcircled{\texttt{k}}$ and $\textcircled{\texttt{e}}$. When we restrict initial evaluation of a formula to models in $\mathcal{C}_\emptyset$ we add $\emptyset$ as a subscript. For example, $\mathcal{ML}^m_\emptyset(\langle\!\langle r \rangle\!\rangle)$ is the modal logic with $\langle\!\langle r \rangle\!\rangle$ instead of $\langle r \rangle$, the operators $\textcircled{\texttt{r}}$ and $\textcircled{\texttt{k}}$, and whose models have an initially empty memory. For the rest of the article, $\mathfrak{L}$ will stand for any memory logic $\mathcal{ML}^m(\ldots)$. For the sake of simplicity we restrict ourselves to the unimodal case. The generalization to the multimodal scenario is straightforward.

2 Bisimulations and Saturated Models

The concept of bisimulation has been extensively studied for many modal logics [6,5]. In the context of memory logics, bisimulations link pairs (A, w) (where $A \cup \{w\}$ is a subset of the domain) between models, as we need to keep track not only of the current state but also of the current memory. Let $\mathcal{M}$ and $\mathcal{N}$ be two memory models. Then a bisimulation between $\mathcal{M}$ and $\mathcal{N}$ is a binary relation such that $(A, w) \sim (B, v)$ implies $A \cup \{w\} \subseteq |\mathcal{M}|$ and $B \cup \{v\} \subseteq |\mathcal{N}|$.

Bisimulations for the different memory logics can be defined modularly. Given a memory logic $\mathfrak{L}$, its bisimulation notion will be defined imposing restrictions to $\sim$ depending on the operators present in $\mathfrak{L}$. In Figure 1 we summarize the restrictions associated with each operator for models $\mathcal{M}$ and $\mathcal{N}$ with accessibility relations R and R' respectively.

always	(nontriv)	$\sim$ is not empty.
always	(agree)	If $(A, w) \sim (B, v)$, then w and v make the same propositional variables true.
ⓚ	(kagree)	If $(A, w) \sim (B, v)$, then $w \in A$ if and only if $v \in B$.
ⓡ	(remember)	If $(A, w) \sim (B, v)$, then $(A \cup \{w\}, w) \sim (B \cup \{v\}, v)$.
ⓕ	(forget)	If $(A, w) \sim (B, v)$, then $(A \setminus \{w\}, w) \sim (B \setminus \{v\}, v)$.
ⓔ	(erase)	If $(A, w) \sim (B, v)$, then $(\emptyset, w) \sim (\emptyset, v)$.
$\langle r \rangle$	(forth)	If $(A, w) \sim (B, v)$ and wRw', then there exists $n' \in \lvert\mathcal{N}\rvert$ such that $vR'v'$ and $(A, w') \sim (B, v')$.
	(back)	If $(A, w) \sim (B, v)$ and $vR'v'$, then there exists $w' \in \lvert\mathcal{M}\rvert$ such that wRw' and $(A, w') \sim (B, v')$.
$\langle\langle r \rangle\rangle$	(mforth)	If $(A, w) \sim (B, v)$ and wRw', then there exists $v' \in \lvert\mathcal{N}\rvert$ such that $vR'v'$ and $(A \cup \{w\}, w') \sim (B \cup \{v\}, v')$.
	(mback)	If $(A, w) \sim (B, v)$ and $vR'v'$, then there exists $w' \in \lvert\mathcal{M}\rvert$ such that wRw' and $(A \cup \{w\}, w') \sim (B \cup \{v\}, v')$.

Fig. 1. Operator restrictions for a modular memory bisimulation definition

With these definitions, we have presented bisimulation notions for all memory logics introduced in Section 1.

If $\mathcal{M}$ is a model and $w \in |\mathcal{M}|$, we call the pair $\langle \mathcal{M}, w \rangle$ a *pointed model.* Given two pointed models $\langle \mathcal{M}, w \rangle$ and $\langle \mathcal{N}, v \rangle$, where $\mathcal{M} = \langle W, R, V, S \rangle$ and $\mathcal{N} = \langle W', R', V', S' \rangle$, we write $\mathcal{M}, w \mathrel{\underline{\leftrightarrow}} \mathcal{N}, v$ if there is a bisimulation linking (S, w) and (S', v). The exact type of bisimulation involved will usually be clear from context; we will write $\mathrel{\underline{\leftrightarrow}}_{\mathfrak{L}}$ when we need to specify that the bisimulation corresponds to the logic $\mathfrak{L}$. We write $\mathcal{M}, w \equiv_{\mathfrak{L}} \mathcal{N}, v$ when both models satisfy the same $\mathfrak{L}$-formulas, i.e., for all $\varphi \in \mathfrak{L}$, $\mathcal{M}, w \models \varphi$ iff $\mathcal{N}, v \models \varphi$. We will again drop the $_{\mathfrak{L}}$ subindex when no confusion arises.

The basic property expected from bisimulation is that they should preserve the satisfiability of formulas. The following theorem states that this is the case for the bisimulations we introduced (see [13] for details).

Theorem 1. *If $\mathcal{M}, w \mathrel{\underline{\leftrightarrow}}_{\mathfrak{L}} \mathcal{N}, v$ then $\mathcal{M}, w \equiv_{\mathfrak{L}} \mathcal{N}, v$.*

With all preliminaries concerning bisimulation already introduced, we now proceed to the notion of ω-saturated models and Hennessy-Milner classes, which will lead to our first result: the class of ω-saturated models is a Hennessy-Milner class for all memory logics we introduced, with respect to the appropriate notion of bisimulation. This property will be fundamental for the results concerning characterization and definability established in the next section.

The notion of ω-saturation [8,9] is defined for first order models, but it also applies to memory models using the correspondence between memory and first order models discussed in Section 1. These models will prove to be a very useful tool. We have already seen that if two states are bisimilar, then they are modally equivalent. The converse, in general, does not hold. We say that a class $\mathcal{C}$ of models *has the Hennessy-Milner* property with respect to $\mathfrak{L}$-bisimulations (or, simply, that the class is Hennessy-Milner for $\mathfrak{L}$) if any two $\mathfrak{L}$-equivalent models in $\mathcal{C}$ are $\mathfrak{L}$-bisimilar. As we will prove in Theorem 4, ω-saturated models are Hennessy-Milner for all memory logics $\mathfrak{L}$.

But ω-saturated models have other important properties, like the '*intra-model* compactness property' enunciated below (the proof is a straightforward modification of the result in [5, Theorem 2.65] for the basic modal logic).

Proposition 2. *Let $\mathcal{M} = \langle W, R, V, S\rangle$ be ω-saturated, Σ be a set of $\mathfrak{L}$-formulas and $w \in W$. If every finite subset $\Delta \subseteq \Sigma$ satisfies $\mathcal{M}, v_\Delta \models \Delta$ for some R-successor v_Δ of w then there exists v, an R-successor of w, such that $\mathcal{M}, v \models \Sigma$.*

It is also the case that ω-saturation is preserved under the operation of memorizing a finite set of elements. The proof can be found in [13].

Proposition 3. *Let $\mathcal{M}$ be ω-saturated. For any finite $A \subseteq |\mathcal{M}|$, $\mathcal{M}[A]$ is ω-saturated.*

Not all models are ω-saturated but a classic theorem of first order logic [8,9] states that every model $\mathcal{M}$ has an ω-saturated extension $\mathcal{M}^+$ with the same first order theory and, a fortiori, the same $\mathfrak{L}$ theory for any memory logic $\mathfrak{L}$. This extension is created by taking an ultrapower of the model with a special kind of ultrafilter.[3]

We now prove that, for every memory logic $\mathfrak{L}$, the class of ω-saturated models has the Hennessy-Milner property with respect to $\mathfrak{L}$-bisimulations.

Theorem 4. *Let $\mathfrak{L}$ be a memory logic, the class of ω-saturated models has the Hennessy-Milner property with respect to $\mathfrak{L}$-bisimulations.*

Proof (Sketch). As we want to consider all the possible logics from the family of memory logics, we prove that, for any two ω-saturated models $\langle \mathcal{M}, w\rangle$ and $\langle \mathcal{N}, v\rangle$ such that $\mathcal{M}, w \equiv_{\mathfrak{L}} \mathcal{N}, v$ there is an $\mathfrak{L}$-bisimulation between them. We do this by considering every possible operator and show that we can construct a bisimulation that satisfies the constraints associated for that operator.

See the Appendix for full details. □

The proof of the theorem above is fairly straightforward, but the result itself is surprising in its generality and can be taken as evidence of a harmonious match between the notion of bisimulation we introduced and the general model theory of memory logics.[4]

[3] In what follows we will assume that the reader is familiar with the definition of ultraproducts, ultrapowers and ultrafilters (consult [11] if necessary).

[4] Notice that a direct corollary of Theorem 4 is that the class of image-finite models (i.e., models where each state has at most a finite number of successors) for any memory logic has the Hennessy-Milner property with respect to $\mathfrak{L}$-bisimulations.

3 Characterization and Definability

While investigating the properties of a new modal logic, a fairly standard approach is to try to characterize it as a fragment of a better known logic. A classical example of this kind of results is van Benthem's characterization of the basic modal logic as the bisimulation invariant fragment of first order logic. These type of characterizations allows for the transfer of results and for a better understanding of the logic. In the following theorem we state an analogous result for memory logics. Due to space limitations we only give a sketch of the proof along with citations that should suffice to complete it.

We say that a first order formula $\alpha(x)$ is *invariant for $\mathfrak{L}$-bimulations* if for all models $\mathcal{M}, \mathcal{N}$ and $w \in |\mathcal{M}|, v \in |\mathcal{N}|$ such that $\mathcal{M}, w \,\underline{\leftrightarrow}_{\mathfrak{L}}\, \mathcal{N}, v$ we have $\mathcal{M}, g^x_w \models \alpha(x)$ iff $\mathcal{N}, g^x_v \models \alpha(x)$.

Theorem 5 (Characterization). *A first order formula $\alpha(x)$ (with free variable x, and in the proper signature) is equivalent to the translation of an $\mathfrak{L}$-formula iff $\alpha(x)$ is invariant for $\mathfrak{L}$-bisimulations.*

Proof (Sketch). The left to right direction is a consequence of Theorem 1. As observed in [7] the main ingredient for the right to left direction is that the class of ω-saturated models have the Hennessy-Milner property. This fact was proved true for the family of memory logics in Theorem 4. The rest of the proof is a routine rephrase of the one found in [5, Theorem 2.68] for the basic modal logic. □

Notice that the result above holds for all the memory logics we introduced.

We now proceed to investigate definability. The study of definability of classes of models – i.e., given an arbitrary logic L which are the classes of models that can be captured as those satisfying a formula (or a set of formulas) of L – is well developed. Results of this kind are well known, for example, in first order logics. Traditionally, a class of models that is definable by means of a set of first order formulas is called *elementary* and those that can be defined by means of a single formula are called *basic elementary* classes.

Definability results for different modal logics have also been established [6,5]. Once more, the results for basic modal logic lifts to memory logics if we consider the appropriate notion of bisimulation.

Theorem 6 (Definability by a set). *A class of pointed models $\mathcal{C}$ is definable by a set of $\mathfrak{L}$-formulas iff $\mathcal{C}$ is closed under $\mathfrak{L}$-bisimulations and under ultraproducts; and the complement of $\mathcal{C}$ is closed under ultrapowers.*

Proof. From left to right. Suppose that $\mathcal{C}$ is defined by the set Γ of $\mathfrak{L}$-formulas and there is a model $\langle \mathcal{M}, w \rangle \in \mathcal{C}$ such that $\mathcal{M}, w \,\underline{\leftrightarrow}\, \mathcal{N}, v$ for some model $\mathcal{N}, v$. As $\langle \mathcal{M}, w \rangle \in \mathcal{C}$ it must occur that $\mathcal{M}, w \models \Gamma$. By bisimulation preservation we have $\mathcal{N}, v \models \Gamma$ therefore $\langle \mathcal{N}, v \rangle \in \mathcal{C}$. Hence, $\mathcal{C}$ is closed under $\mathfrak{L}$-bisimulations.

If $\mathcal{C}$ is definable by a set Γ of $\mathfrak{L}$-formulas it is also defined by the first order translation of Γ. Therefore $\mathcal{C}$ is elementary which implies that it is closed under ultraproducts and its complement is closed under ultrapowers [8,9].

From right to left. Suppose $\mathcal{C}$ is closed under $\mathfrak{L}$-bisimulations and ultraproducts, while its complement is closed under ultrapowers. Let Γ be the set of $\mathfrak{L}$-formulas true in every model of $\mathcal{C}$. Trivially $\mathcal{C} \models \Gamma$. We still have to show that if $\mathcal{M}, w \models \Gamma$ then $\langle \mathcal{M}, w \rangle \in \mathcal{C}$. Define the following set

$$\mathsf{Th}^{w}(x) = \{\mathsf{ST}_x(\varphi) : \varphi \text{ is an } \mathfrak{L}\text{-formula and } \mathcal{M}, w \models \varphi\}.$$

We state that $\mathsf{Th}^{w}(x)$ is satisfiable in $\mathcal{C}$. For suppose not. By compactness, there is a finite subset $\Sigma_0 \subseteq \mathsf{Th}^{w}(x)$ such that $\Sigma_0 = \{\sigma_1, \ldots, \sigma_n\}$ is not satisfiable in $\mathcal{C}$.[5] This means that the formula $\psi = \neg \bigwedge_i \sigma_i$ is valid in $\mathcal{C}$ and therefore $\psi \in \Gamma$. This is a contradiction because it is obvious that $\mathcal{M}, w \not\models \psi$ and by hypothesis $\mathcal{M}, w \models \Gamma$. Hence, there is a model $\langle \mathcal{N}, v \rangle \in \mathsf{K}$ such that $\mathcal{N}, v \models \mathsf{Th}^{w}(x)$. It is easy to see that these models satisfy $\mathcal{N}, v \equiv_{\mathfrak{L}} \mathcal{M}, w$.

To finish, suppose that $\langle \mathcal{M}, w \rangle \notin \mathcal{C}$, we take ω-saturated extensions $\langle \mathcal{N}^*, v^* \rangle \in \mathcal{C}$ and $\langle \mathcal{M}^*, w^* \rangle \notin \mathcal{C}$. As ω-saturated models have the Hennessy-Milner property (by Theorem 4) this implies that $\mathcal{N}^*, v^* \leftrightarroweq_{\mathfrak{L}} \mathcal{M}^*, w^*$. As $\mathcal{C}$ is closed under bisimulations then $\langle \mathcal{M}, w \rangle \in \mathcal{C}$, a contradiction. Therefore $\langle \mathcal{M}, w \rangle$ must be in $\mathcal{C}$. □

Theorem 7 (Definability by a single formula). *A class of pointed models $\mathcal{C}$ is definable by a single $\mathfrak{L}$-formula iff $\mathcal{C}$ is closed under $\mathfrak{L}$-bisimulations and both $\mathcal{C}$ and its complement are closed under ultraproducts.*

Proof. From left to right. Suppose $\mathcal{C}$ is definable by a single $\mathfrak{L}$-formula φ. Observe that the complement of $\mathcal{C}$ is defined by $\neg\varphi$. Using Theorem 6 on $\mathcal{C}$ with $\Gamma = \{\varphi\}$ and on its complement with $\Gamma = \{\neg\varphi\}$ we conclude what we wanted to prove.

From right to left. Suppose that $\mathcal{C}$ is closed under $\mathfrak{L}$-bisimulations and both $\mathcal{C}$ and its complement are closed under ultraproducts. As the bisimulation relation is symmetric it is easy to see that $\mathcal{C}$ is closed under bisimulations iff its complement is. Using this fact and Theorem 6 twice we have sets of formulas Γ_1 defining $\mathcal{C}$ and Γ_2 defining its complement.

It is obvious that the union of these sets cannot be consistent. Therefore, by compactness, there exist $\{\alpha_1, \ldots, \alpha_n\} \subseteq \Gamma_1$ and $\{\beta_1, \ldots, \beta_m\} \subseteq \Gamma_2$ such that $\bigwedge_i \alpha_i \to \neg \bigwedge_j \beta_j$ is valid. We claim that it is exactly $\varphi = \bigwedge_i \alpha_i$ that defines $\mathcal{C}$.

If $\langle \mathcal{M}, w \rangle \in \mathcal{C}$, it satisfies Γ_1 and in particular φ. Suppose that $\mathcal{M}, w \models \varphi$. Hence $\mathcal{M}, w \models \neg \bigwedge_j \beta_j$ and therefore $\mathcal{M}, w \not\models \Gamma_2$; i.e., $\langle \mathcal{M}, w \rangle \in \mathcal{C}$. □

As further research it would be interesting to investigate if Theorems 6 and 7 can be restated using closure under *ultrafilter unions*, as defined in [15].

4 Interpolation

The notion of bisimulation also plays a crucial role for proving and disproving interpolation properties. Given a formula φ, let $\mathsf{props}(\varphi)$ be the set of propositional symbols occurring in φ. A modal logic has *interpolation over propositional symbols* on a class $\mathcal{C}$, if for all formulas φ, ψ such that $\mathcal{C} \models \varphi \to \psi$, there is a modal

[5] The compactness theorem preserves ultraproducts-closed classes (see [7]).

formula δ (usually called the *interpolant*) such that $\mathcal{C} \models \varphi \to \delta$, $\mathcal{C} \models \delta \to \psi$, and $\mathsf{props}(\delta) \subseteq \mathsf{props}(\varphi) \cap \mathsf{props}(\psi)$. Note that there is no restriction on the modalities occurring in δ.

We will show that most of the memory logics we are studying lack interpolation (with the exception of $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ and $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle, \textcircled{f})$). We will use a classic technique to prove this, whose general schema is the following. First, we define φ and ψ such that $\varphi \to \psi$ is a valid formula. Then, we find two models $\langle \mathcal{M}, w \rangle$ and $\langle \mathcal{M}', w' \rangle$, such that w and w' are bisimilar in the common language of φ and ψ, but $\mathcal{M}, w \models \varphi$ while $\mathcal{M}', w' \models \neg\psi$. This is enough to claim that interpolation fails. For suppose that interpolation holds. Then there is an interpolant δ in the common language of φ and ψ such that $\varphi \to \delta$ and $\delta \to \psi$ are valid. Therefore δ holds at $\langle \mathcal{M}, w \rangle$. Because w and w' are bisimilar in the common language, δ also holds at $\langle \mathcal{M}', w' \rangle$. This implies that ψ holds at $\langle \mathcal{M}', w' \rangle$ too, but this is a contradiction, since we assumed that $\neg\psi$ holds there.

In the context of memory logics, there is a choice to make concerning the inclusion of $\textcircled{k}$ in the common language. We will use the term *interpolation over propositional symbols and* $\textcircled{k}$ when we decide to include it in the common language. We will just say "the common language" when no confusion between the two notions can arise. Observe that $\top$ can always occur in the interpolant, since otherwise the definition of interpolation can be easily trivialized. Finally, unless we explicitly say otherwise, we prove interpolation (or the lack thereof) for the class of all models.

We first show that interpolation over propositional symbols fails for $\mathcal{ML}^m(\langle r \rangle)$ and its extension with the $\textcircled{e}$ operator. This is also true for some fragments that use $\langle\!\langle r \rangle\!\rangle$ instead of $\langle r \rangle$, both over the class of all models and over $\mathcal{C}_\emptyset$.

Theorem 8. *The logics* $\mathcal{ML}^m(\langle r \rangle)$, $\mathcal{ML}^m(\langle r \rangle, \textcircled{e})$, $\mathcal{ML}^m_\emptyset(\langle r \rangle)$, $\mathcal{ML}^m_\emptyset(\langle\!\langle r \rangle\!\rangle)$ *and* $\mathcal{ML}^m_\emptyset(\langle r \rangle, \textcircled{e})$ *lack interpolation over propositional symbols.*

Proof (Sketch). Full details are given in the Appendix. The key ingredient of the proofs for each logic is the ability to find two models which are bisimilar in the common language. These results are strongly based on bisimilar models used in [1] to investigate relative expressive power of different memory logics. □

We leave the analysis for $\textcircled{f}$ open, since we could not find an equivalent pair of models for this case. See [13] for more details.

Now we show that $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ has interpolation over propositional symbols and $\textcircled{k}$ with respect to a quite general class of models. The technique we use here is similar to the one presented in [14]. To develop the proof we will need some tools from model theory. We introduce some definitions and preliminary results and refer the reader to [8,9,14,12] for details.

Throughout the rest of this section, $\underline{\leftrightarrow}$ refers to $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-bisimulation. We will use $\underline{\leftrightarrow}_{\mathcal{ML}}$ when we want to refer to $\mathcal{ML}$-bisimulations.

Definition 9. *1. A* total $\mathcal{ML}$ frame bisimulation *between frames* $\langle W, R \rangle$ *and* $\langle W', R' \rangle$ *is a total binary relation on* $W \times W'$ *satisfying conditions (nontriv), (forth) and (back) from Figure 1.*

2. *An $\mathcal{ML}$-*bisimulation product *of a set of frames $\{\mathcal{F}_i \mid i \in I\}$ is a subframe $\mathcal{B}$ of the cartesian product $\Pi_i\mathcal{F}_i$ such that for each $i \in I$, the natural projection function $f_i : \mathcal{B} \to \mathcal{F}_i$ is a surjective bounded morphism.*

Bisimulation producs, together with the following theorem (see [12] for the proof), allow us to construct a new frame using a total $\mathcal{ML}$ frame bisimulation between two given frames. This will be helpful later to construct a model that will act as a witness for the interpolant.

Theorem 10. *Let $\mathcal{H}$ be a subframe of the product $\mathcal{F} \times \mathcal{G}$. Then $\mathcal{H}$ is an $\mathcal{ML}$-bisimulation product of $\mathcal{F}$ and $\mathcal{G}$ iff the domain of $\mathcal{H}$ is a total $\mathcal{ML}$ frame bisimulation between $\mathcal{F}$ and $\mathcal{G}$.*

The last ingredient we need is to define total bisimulation in the context of memory logics. Intuitively, it is a bisimulation in which every possible relevant pairs are related.

Definition 11 (Total $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-bisimulation). *Let $\mathcal{M} = \langle W, R, V, S\rangle$ and $\mathcal{N} = \langle W', R', V', S'\rangle$ be two models of $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$. We say that $\mathcal{M}, w$ and $\mathcal{N}, v$ are* totally bisimilar *($\mathcal{M}, w \underline{\leftrightarrow}^T \mathcal{N}, v$) when there is bisimulation $\sim$ between $\mathcal{M}, w$ and $\mathcal{N}, v$ and*

1. *for every $A = \{a_1, \ldots, a_k\} \subseteq W$ with $a_i R a_{i+1}$, and every $a \in W$ there is a $B = \{b_1, \ldots, b_k\} \subseteq W'$ with $b_i R' b_{i+1}$ and $b \in W'$ such that $(A, a) \sim (B, b)$*
2. *for every $B = \{b_1, \ldots, b_k\} \subseteq W'$, and every $b \in W'$ there is $A = \{a_1, \ldots, a_k\} \subseteq W$ with $a_i R a_{i+1}$ and $a \in M$ such that $(A, a) \sim (B, b)$.*

Theorem 12. *Let $\mathcal{C}$ be any elementary frame class closed under generated subframes and bisimulation products. Then $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ and $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle, \textcircled{f})$ have interpolation over propositions and $\textcircled{k}$ relative to the class of all models with frame in $\mathcal{C}$.*

Proof (Sketch). Full details are provided in the Appendix. Suppose there are two $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-formulas φ and ψ such that $\varphi \to \psi$ is valid, but it does not have an interpolant in the common language. In general, the bisimulations we discuss here between a pair of models are always established with respect to the common language of φ and ψ. We first show that there are two models $\mathcal{M}$ and $\mathcal{N}$ such that $\mathcal{M}, w \models \varphi$ and $\mathcal{N}, v \models \neg\psi$. We next take ω-saturated models $\mathcal{M}^+$ and $\mathcal{N}^+$ of $\mathcal{M}$ and $\mathcal{N}$ respectively and show $\mathcal{M}^+, w \underline{\leftrightarrow}^T \mathcal{N}^+, v$. According to the tree model property for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ (see [1]), we take equivalent tree $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-models $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ such that $\mathcal{M}^+, w \underline{\leftrightarrow}^T \mathcal{M}^+_T, w$ and $\mathcal{N}^+, v \underline{\leftrightarrow}^T \mathcal{N}^+_T, v$. We conclude $\mathcal{M}^+_T, w \underline{\leftrightarrow}^T \mathcal{N}^+_T, v$.

Then we switch to the basic modal logic $\mathcal{ML}$. Let $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ be the corresponding $\mathcal{ML}$-models of $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ respectively (shifting the signature to $\langle \textsc{prop} \cup \{known\}, \textsc{rel}\rangle$). Being $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ an extension of $\mathcal{ML}$, we have $\mathcal{M}^+_{T_{\mathcal{ML}}} \underline{\leftrightarrow}^T_{\mathcal{ML}} \mathcal{N}^+_{T_{\mathcal{ML}}}$. Using Theorem 10, one can show that there is a bisimulation product $\mathcal{H} \in \mathcal{C}$ of the frames of $\mathcal{M}'$ and $\mathcal{N}'$, and a valuation V such that $(\mathcal{H}, V), \langle w, v\rangle \models (\varphi \wedge \neg\psi)[\textcircled{k}/known]$.

Since by its definition, $\mathcal{H}$ is a tree, we can return to $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ and conclude that $\varphi \wedge \neg\psi$ is satisfiable in some $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-model based on a frame in $\mathcal{C}$, contradicting our hypothesis. Graphically, the general schema is the following (the double headed arrows represent total bisimulations):

$$\begin{tikzcd}
\mathcal{N} \arrow[r, leftrightarrow] \arrow[dd, leftrightarrow] & \mathcal{N}^+ \arrow[r, leftrightarrow] \arrow[dd, leftrightarrow] & \mathcal{N}^+_T \arrow[dd, leftrightarrow] & \equiv & \mathcal{N}^+_{T_{\mathcal{ML}}} \arrow[dd, leftrightarrow, "\mathcal{ML}"] \arrow[dr, leftrightarrow, bend left, "\mathcal{ML}"] & \\
 & & & & & (\mathcal{H}, V) \\
\mathcal{M} \arrow[r, leftrightarrow] & \mathcal{M}^+ \arrow[r, leftrightarrow] & \mathcal{M}^+_T & \equiv & \mathcal{M}^+_{T_{\mathcal{ML}}} \arrow[ur, leftrightarrow, bend right, "\mathcal{ML}"'] &
\end{tikzcd}$$

Following this schema the result can be proved for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$. Then, interpolation for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle, \textcircled{f})$ is straightforward using the equivalence preserving translations defined in [1] between $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle, \textcircled{f})$ and $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$. □

5 Conclusions and Further Work

In this article we investigated some model theoretical properties of several memory logics. Fist we analyzed memory logics in terms of first-order characterization and definability. These properties hold for all the logics we introduced, thanks to a general Hennessy-Milner property for ω-saturated models. Then we studied interpolation and showed that the property fails for many memory logics both over the class of all models and over $\mathcal{C}_\emptyset$. On the other hand, we stablished interpolation over propositional symbols and known for $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$ and $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle, \textcircled{f})$ over many different classes of models. Bisimulations were a key tool to tackle these problems. The results presented here help complete a picture of the properties of memory logics and contributes to understanding what they are, how they behave, and which is their relation with other well-known logics.

There are some pending problems that are worth investigating. The expressive power of some memory logics is still not well understood (in particular, when the language includes the $\textcircled{f}$ operator, see [1]). This directly leads to the still unanswered questions concerning interpolation. Also, the Beth definability property is usually studied together with interpolation, and for many logics, a proof of the former can be obtained once a proof of the later is at hand. Both properties are closely connected, and the logics having one but not the other are relatively few (see [10] for examples). Alas! the case for memory logics is not that simple. Even though some weak results concerning Beth definability for memory logics have been established (see [13]), a general conclusion is still missing.

References

1. Areces, C., Figueira, D., Figueira, S., Mera, S.: The expressive power of memory logics. Review of Symbolic Logic 4(1) (2010)
2. Areces, C., Figueira, D., Gorín, D., Mera, S.: Tableaux and model checking for memory logics. In: Giese, M., Waaler, A. (eds.) TABLEAUX 2009. LNCS, vol. 5607, pp. 47–61. Springer, Heidelberg (2009)

3. Areces, C., Figueira, S., Mera, S.: Completeness results for memory logics. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 16–30. Springer, Heidelberg (2008)
4. Areces, C., Figueira, D., Figueira, S., Mera, S.: Expressive power and decidability for memory logics. In: Hodges, W., de Queiroz, R. (eds.) WoLLIC 2008. LNCS (LNAI), vol. 5110, pp. 56–68. Springer, Heidelberg (2008)
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
6. Blackburn, P., Wolter, F., van Benthem, J. (eds.): Handbook of Modal Logics. Elsevier, Amsterdam (2006)
7. Carreiro, F.: Characterization and definability in modal first-order fragments. Master's thesis, Universidad de Buenos Aires, arXiv:1011.4718 (2010)
8. Chang, C., Keisler, H.: Model Theory, Studies in Logic and the Foundations of Mathematics, 3rd edn., vol. 73. North-Holland Publishing Co., Amsterdam (1990)
9. Doets, K.: Basic model theory. University of Chicago Press, Chicago (1996)
10. Hoogland, E.: Definability and interpolation: Model-theoretic investigations. Ph.D. thesis, ILLC. Universiteit van Amsterdam (2001)
11. Keisler, H.J.: The ultraproduct construction. In: Proceedings of the Ultramath Conference, Pisa, Italy (2008)
12. Marx, M., Venema, Y.: Multi-dimensional modal logic. Kluwer, Dordrecht (1997)
13. Mera, S.: Modal Memory Logics. Ph.D. thesis, Universidad de Buenos Aires & Université Henri Poincare, Buenos Aires, Argentina (2009)
14. ten Cate, B.: Model theory for extended modal languages. Ph.D. thesis, University of Amsterdam, ILLC Publications, Ph. D. Dissertation series, Amsterdam (2005)
15. Venema, Y.: Ultrafilter unions: an exercise in modal definability. In: First Workshop on Logic and Language, pp. 303–310. Universidad de Sevilla (2000)

Appendix

Theorem 4. *Let $\mathfrak{L}$ be a memory logic, the class of ω-saturated models has the Hennessy-Milner property with respect to $\mathfrak{L}$-bisimulations.*

Proof. Given two ω-saturated models $\mathcal{M} = \langle W, R, V, S\rangle$ and $\mathcal{N} = \langle W', R', V', S'\rangle$ we propose the binary relation $\sim$ defined as

$$(A, w) \sim (B, v) \ \text{ iff } \ \mathcal{M}', w \equiv_{\mathfrak{L}} \mathcal{N}', v$$

as a candidate for a bisimulation where $\mathcal{M}' = \langle W, R, V, A\rangle$, $\mathcal{N}' = \langle W', R', V', B\rangle$ and $A \cup \{w\} \subseteq W$, $B \cup \{v\} \subseteq W'$. Suppose that $(A, w) \sim (B, v)$. $\sim$ satisfies *(nontriv)* and *(agree)* by definition.

(kagree): If ⓚ is an operator of $\mathfrak{L}$, then $w \in A$ iff $\mathcal{M}', w \models$ ⓚ iff $\mathcal{N}', v \models$ ⓚ iff $v \in B$. This proves that *(kagree)* is satisfied.

(remember): Suppose that ⓡ is an operator of $\mathfrak{L}$. Then $(A, w) \sim (B, v)$ implies that for every φ, $\mathcal{M}', w \models \varphi$ iff $\mathcal{N}', v \models \varphi$. In particular, $\mathcal{M}', w \models$ ⓡψ iff $\mathcal{N}', v \models$ ⓡψ which by satisfaction definition holds precisely when $\mathcal{M}'[w], w \models \psi$ iff $\mathcal{N}'[v], v \models \psi$ and hence $(A \cup \{w\}, w) \sim (B \cup \{v\}, v)$. This proves that *(remember)* is satisfied. The conditions *(forget)* and *(erase)* are stablished similarly in logics with the ⓕ and ⓔ operators.

(forth) and *(back)*: These properties are proved as for basic modal logic (see [5, Proposition 2.54]).

(mforth) and *(mback)*: Since $(A, w) \sim (B, v)$, we have already seen in the $\textcircled{r}$ case that $\mathcal{M}'[w], w \models \psi$ iff $\mathcal{N}'[v], v \models \psi$.[6] This implies that $\mathcal{M}'[w], w \equiv_{\mathfrak{L}} \mathcal{N}'[v], v$. Using Lemma 3 we also know that $\langle \mathcal{M}'[w], w\rangle$ and $\langle \mathcal{N}'[v], v\rangle$ are both ω-saturated.

Suppose that w' is a successor of w. Let Σ be the set of all the formulas true at $\mathcal{M}'[w], w'$. For every finite subset $\Delta \subseteq \Sigma$ we have $\mathcal{M}'[w], w' \models \bigwedge \Delta$ and therefore $\mathcal{M}'[w], w \models \langle\!\langle r \rangle\!\rangle \bigwedge \Delta$. By $\mathfrak{L}$-equivalence we have $\mathcal{N}'[v], v \models \langle\!\langle r \rangle\!\rangle \bigwedge \Delta$ which means that for every Δ we have a v-successor which satisfies it. By Lemma 2 we can conclude that there exists v' a v-successor so that $\mathcal{N}'[v], v' \models \Sigma$.

As $\mathcal{M}'[w], w'$ and $\mathcal{N}'[v], v'$ make the same formulas true, then they are $\mathfrak{L}$-equivalent and by definition they will be related by the bisimulation. This proves that *(mforth)* is satisfied because $(A \cup \{w\}, w') \sim (B \cup \{v\}, v')$. The proof for *(mback)* is similar but switching the models. □

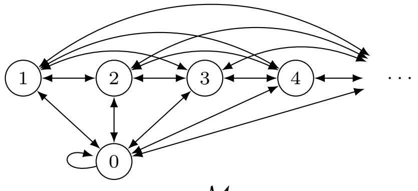
Theorem 8. *The logics* $\mathcal{ML}^m(\langle r \rangle)$, $\mathcal{ML}^m(\langle r \rangle, \textcircled{e})$, $\mathcal{ML}^m_{\emptyset}(\langle r \rangle)$, $\mathcal{ML}^m_{\emptyset}(\langle\!\langle r \rangle\!\rangle)$ *and* $\mathcal{ML}^m_{\emptyset}(\langle r \rangle, \textcircled{e})$ *lack interpolation over propositional symbols.*

Proof. We show each case separately.

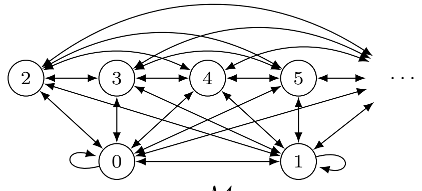
$\mathcal{ML}^m_{\emptyset}(\langle r \rangle)$: Let $\varphi = q \wedge \textcircled{r}[r](\neg \textcircled{k} \rightarrow \varphi')$. If $\mathcal{M}, w \models \varphi$ then q is true at w and any successor of w different from w satisfies φ'. Now, let $\varphi' = \neg q \wedge \neg \textcircled{r}\langle r \rangle(\textcircled{k} \wedge \neg q)$. With this definition of φ', if $\mathcal{M}, w \models \varphi$ then for all v such that wRv and $v \neq w$ we have $\neg vRv$.

Let $\psi = p \wedge \langle r \rangle(\neg p \wedge \textcircled{r}\langle r \rangle\textcircled{k})$. If $\mathcal{M}, w \models \psi$ then there is $v \neq w$ such that wRv and vRv. It is clear that $\varphi \wedge \psi$ is a contradiction, so $\varphi \rightarrow \neg\psi$ is valid.

Let $\mathcal{M}_1 = \langle \mathbb{N}, R_1, \emptyset, \emptyset \rangle$ and $\mathcal{M}_2 = \langle \mathbb{N}, R_2, \emptyset, \emptyset \rangle$, where $R_1 = \{(n, m) \mid n \neq m\} \cup \{(0, 0)\}$ and $R_2 = R_1 \cup \{(1, 1)\}$. Graphically,



$\mathcal{M}_1$



$\mathcal{M}_2$

where the accessibility relation is the transitive closure of the arrows shown but without reflexive loops excepts those explicitly marked. In [1] it was shown that $\langle \mathcal{M}_1, 0\rangle$ and $\langle \mathcal{M}_2, 0\rangle$ are bisimilar over $\mathcal{ML}^m_{\emptyset}(\langle r \rangle)$. Now, define the models $\mathcal{M}'_1$ and $\mathcal{M}'_2$ as $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively but with a nonempty valuation in the following way: $\mathcal{M}'_1 = \langle \mathbb{N}, R_1, V_1, \emptyset \rangle$ and $\mathcal{M}'_2 = \langle \mathbb{N}, R_2, V_2, \emptyset \rangle$, where $R_1 = \{(n, m) \mid n \neq m\} \cup \{(0, 0)\}$, $R_2 = R_1 \cup \{(1, 1)\}$, $V_1(q) = \{0\}$ and $V_2(p) = \{0\}$. One can verify that $\langle \mathcal{M}'_1, 0\rangle$ and $\langle \mathcal{M}'_2, 0\rangle$ are bisimilar over the common language and that $\mathcal{M}'_1, 0 \models \varphi$ and $\mathcal{M}'_2, 0 \models \psi$.

[6] We can use $\textcircled{r}$ here because we required that every memory logic should have it.

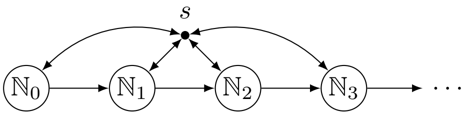
Suppose there is an interpolant χ over the common language of φ and ψ for the valid formula $\varphi \to \neg\psi$. On the one hand, since φ is true at $\langle \mathcal{M}'_1, 0\rangle$ then χ also is. On the other, since ψ is true at $\langle \mathcal{M}'_2, 0\rangle$ then $\neg\chi$ also is. Then we have that $\mathcal{M}'_1, 0 \models \chi$ and $\mathcal{M}'_2, 0 \models \neg\chi$, which is a contradiction because $\langle \mathcal{M}'_1, 0\rangle$ and $\langle \mathcal{M}'_2, 0\rangle$ are bisimilar over the common language.

$\mathcal{ML}^m(\langle r\rangle)$: Let φ and ψ be as in the case for $\mathcal{ML}^m_\emptyset(\langle r\rangle)$. Let $\theta = \neg\textcircled{k} \wedge [r]\neg\textcircled{k} \wedge [r][r]\neg\textcircled{k}$. Define $\varphi' = \varphi \wedge \theta$ and $\psi' = \psi \wedge \theta$ and repeat the proof above.

$\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$: Observe that in the proof for $\mathcal{ML}^m_\emptyset(\langle r\rangle)$, instead of ψ, one could use $\psi' = p \wedge \textcircled{r}\langle r\rangle(\neg p \wedge \textcircled{r}\langle r\rangle(\textcircled{k} \wedge \neg p))$. Now, in both φ and ψ', all occurrences of $\langle r\rangle$ are of the form $\textcircled{r}\langle r\rangle$, and all occurrences of $[r]$ are of the form $\textcircled{r}[r]$. Therefore they can be translated to $\langle\langle r\rangle\rangle$ and $[\![r]\!]$ preserving equivalence. Since $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$ is less expressive than $\mathcal{ML}^m_\emptyset(\langle r\rangle)$, both models of the proof for $\mathcal{ML}^m_\emptyset(\langle r\rangle)$ are $\mathcal{ML}^m_\emptyset(\langle\langle r\rangle\rangle)$-bisimilar and therefore the argument is valid.

$\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$: Let $\theta(q) = \textcircled{r}\langle r\rangle(q \wedge \textcircled{k} \wedge \neg\langle r\rangle(\neg q \wedge \textcircled{k}))$. Suppose $\mathcal{M}$ is a model with $S = \{w\}$ where $\mathcal{M}, w \models q$ and $\mathcal{M}, v \models \neg q$. It is not difficult to see that $\mathcal{M}, v \models \theta(q)$ iff vRw and $\neg wRv$. Now, let $\varphi = q \wedge \textcircled{r}\langle r\rangle\langle r\rangle(\neg q \wedge \theta(q))$ and $\psi = p \wedge \textcircled{r}[r][r](\neg\textcircled{k} \to (\neg p \wedge \neg\theta(p)))$ (here $\theta(p)$ is the result of replacing all occurrences of q by p in the formula $\theta(q)$). If φ is true at a point w then there are points u and $v \neq w$ such that $wRuRv$ and vRw and $\neg wRv$. If ψ is true at a point w then for all points u and $v \neq w$ such that $wRuRv$ it is not the case that and vRw and $\neg wRv$. Hence $\models \varphi \to \neg\psi$.

Let $\mathcal{M} = \langle \{s\} \cup \mathbb{N}_0 \cup \mathbb{N}_1 \cup \ldots, R, \emptyset, \emptyset\rangle$, where each $\mathbb{N}_i$ is a different copy of $\mathbb{N}$, and $R = \{(n, m) \mid n \in \mathbb{N}_i, m \in \mathbb{N}_j, i \leq j\} \cup \{(n, s), (s, n) \mid \text{for all } n \neq s\}$. Graphically,



In [1] it was showed that $\langle \mathcal{M}, w_0\rangle$ and $\langle \mathcal{M}, w_1\rangle$ are $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$-bisimilar, where $w_0 \in \mathbb{N}_0$ and $w_1 \in \mathbb{N}_1$. Let $\mathcal{M}'$ be as $\mathcal{M}$ but with a nonempty valuation: $V(p) = \{w_0\}$, $V(q) = \{w_1\}$, and $V(r) = \emptyset$ for all $r \in \textsc{prop}$ different from p and q. It is straightforward to verify that $\mathcal{M}', w_0 \models \psi$ and $\mathcal{M}', w_1 \models \varphi$, but $\langle \mathcal{M}', w_0\rangle$ and $\langle \mathcal{M}', w_1\rangle$ are $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$-bisimilar in the common language.

$\mathcal{ML}^m(\langle r\rangle, \textcircled{e})$: Let φ and ψ be as in the proof for $\mathcal{ML}^m_\emptyset(\langle r\rangle, \textcircled{e})$. It is easy to see that $\textcircled{e}\varphi \to \neg\textcircled{e}\psi$ is a valid formula in the class of $\mathcal{ML}^m(\langle r\rangle, \textcircled{e})$-models. The rest of the argument is similar. □

Theorem 12. *Let $\mathcal{C}$ be any elementary frame class closed under generated subframes and bisimulation products. Then $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ and $\mathcal{ML}^m(\langle\langle r\rangle\rangle, \textcircled{f})$ have interpolation over propositions and known relative to the class of all models with frame in $\mathcal{C}$.*

Proof. We only give the proof of the main theorem. The proofs for the auxiliary lemmas can be found in [13]. Let φ and ψ such that $\mathcal{C} \models \varphi \to \psi$ and let $\mathcal{L}$ be the common language of φ and ψ. Suppose for the sake of contradiction that there is no interpolant of φ and ψ in the language $\mathcal{L}$. We first state two easy lemmas:

Lemma 13. *There is a model $\mathcal{M}$ based on a frame in $\mathcal{C}$, with a state w, such that $\mathcal{M}, w \models \{\chi \mid \mathcal{C} \models \varphi \to \chi$ and $\chi \in \mathcal{L}\} \cup \{\neg\psi\}$.*

Since $\mathcal{C}$ is closed under generated subframes we may assume that $\mathcal{M}$ is generated by w.

Lemma 14. *There is a model $\mathcal{N}$ based on a frame in $\mathcal{C}$, with a state v, such that $\mathcal{N}, v \models \{\chi \mid \mathcal{M}, w \models \chi$ and $\chi \in \mathcal{L}\} \cup \{\varphi\}$.*

Again we may assume that $\mathcal{N}$ is generated by v. Let $\mathcal{M}^+$ and $\mathcal{N}^+$ be ω-saturated elementary extensions of $\mathcal{M}$ and $\mathcal{N}$ respectively. Let us suppose that the first order models $\mathcal{M}^+$ and $\mathcal{N}^+$ have domains M and N and binary relations R_1 and R_2 for the modal operator $\langle r \rangle$, respectively.

We define the relation $\sim$ between $\wp(M) \times M$ and $\wp(N) \times N$ in the following way: for all finite $A \subseteq M$ and finite $B \subseteq N$,

$$(A, a) \sim (B, b) \text{ iff for all formulas } \chi \text{ in } \mathcal{L},\ \mathcal{M}^+[A], a \models \chi \text{ iff } \mathcal{N}^+[B], b \models \chi.$$

By construction $(\emptyset, w) \sim (\emptyset, v)$. We prove that $\sim$ is a bisimulation. Call ST_x the translation from $\mathcal{ML}^m(\langle r \rangle)$ formulas to first order logic formulas defined in [1].

Lemma 15. *$\sim$ is an $\mathcal{ML}^m(\langle\!\langle r \rangle\!\rangle)$-bisimulation between $\mathcal{M}^+$ and $\mathcal{N}^+$ with respect to $\mathcal{L}$.*

Proof. By the definition of $\sim$, it is clear that the condition (*agree*) is satisfied, restricted to $\mathcal{L}$. Let us see (*mzig*). Suppose $(A, a) \sim (B, b)$ and aR_1a'. Let

$$\Gamma = \{\mathsf{ST}_x(\chi) \mid \mathcal{M}^+[A \cup \{a\}], a' \models \chi \text{ and } \chi \in \mathcal{L}\}.$$

Let c_b be a new constant denoting the element b of $\mathcal{N}^+$. We next show that $\Gamma \cup \{R(c_b, x)\}$ is realized in $\mathcal{N}^+[B \cup \{b\}]$, where R is the first order binary relation symbol for $\langle\!\langle r \rangle\!\rangle$. Since, by Lemma 3, the expansion of $\mathcal{N}^+[B \cup \{b\}]$ with the constant c_b is 1-saturated, it suffices to show that every finite subset of Γ is realized in $\mathcal{N}^+[B \cup \{b\}]$ by an R_2-successor of b. Let $\mathsf{ST}_x(\chi_1), \ldots, \mathsf{ST}_x(\chi_n) \in \Gamma$. We have $\mathcal{M}^+[A], a \models \langle\!\langle r \rangle\!\rangle(\chi_1 \wedge \cdots \wedge \chi_n)$, and therefore $\mathcal{N}^+[B], b \models \langle\!\langle r \rangle\!\rangle(\chi_1 \wedge \cdots \wedge \chi_n)$, which implies that there is an R_2-successor of b which satisfies $\chi_1 \wedge \cdots \wedge \chi_n$. I.e., in $\mathcal{N}^+[B \cup \{b\}]$ there is an R_2-successor which realizes $\{\mathsf{ST}_x(\chi_1), \ldots, \mathsf{ST}_x(\chi_n)\}$. Hence, there is b', bR_2b' such that $\mathcal{N}^+[B \cup \{b\}], g^x_{b'} \models \Gamma$. Therefore for every χ of $\mathcal{L}$, if $\mathcal{M}^+[A \cup \{a\}], a' \models \chi$ then $\mathcal{N}^+[B \cup \{b\}], b' \models \chi$. To see the other implication, suppose by contradiction that $\mathcal{N}^+[B \cup \{b\}], b' \models \chi$ but $\mathcal{M}^+[A \cup \{a\}], a' \not\models \chi$ (the case $\mathcal{M}^+[A \cup \{a\}], a' \models \chi$ but $\mathcal{N}^+[B \cup \{b\}], b' \not\models \chi$ is similar). This would imply that $\mathcal{M}^+[A \cup \{a\}], a' \models \neg\chi$ and hence $\mathcal{N}^+[B \cup \{b\}], b' \models \neg\chi$ which leads to a contradiction. The (*mzag*) condition is similar.

In order to check (*remember*), suppose that $\mathcal{M}^+[A], a \models \chi$ iff $\mathcal{N}^+[B], b \models \chi$ for all χ of $\mathcal{L}$. Now, let χ be any formula of $\mathcal{L}$. By hypothesis, $\mathcal{M}^+[A], a \models ⓡ\chi$ iff $\mathcal{N}^+[B], b \models ⓡ\chi$. Applying the definition of ⓡ, we obtain $\mathcal{M}^+[A \cup \{a\}], a \models \chi$ iff $\mathcal{N}^+[B \cup \{b\}], b \models \chi$. □

The following lemma helps prove that $\sim$ is total.

Lemma 16. *For every $a \in M$ there is $b \in N$ such that $(\emptyset, a) \sim (\emptyset, b)$; also for every $b \in N$ there is $a \in M$ such that $(\emptyset, a) \sim (\emptyset, b)$*

Corollary 17. *The $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation $\sim$ is total.*

Applying the tree model property for $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$ (see [1]), let $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$ be tree $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-models such that $\mathcal{M}^+, w \leftrightarroweq^T \mathcal{M}^+_T, w$ and $\mathcal{N}^+, v \leftrightarroweq^T \mathcal{N}^+_T, v$. By Corollary 17, $\mathcal{M}^+, w \leftrightarroweq^T \mathcal{N}^+, v$, and by transitivity of total bisimulations, we conclude $\mathcal{M}^+_T, w \leftrightarroweq^T \mathcal{N}^+_T, v$.

Now, let $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ be the $\mathcal{ML}$ equivalent models for $\mathcal{M}^+_T$ and $\mathcal{N}^+_T$. Since $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation implies $\mathcal{ML}$-bisimulation, $\mathcal{M}^+_{T_{\mathcal{ML}}} \leftrightarroweq^T_{\mathcal{ML}} \mathcal{N}^+_{T_{\mathcal{ML}}}$. Let $\mathcal{F}$ and $\mathcal{G}$ be the underlying frames of $\mathcal{M}^+_{T_{\mathcal{ML}}}$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ respectively. Using Theorem 10, we know there is a bisimulation product $\mathcal{H} \in \mathcal{C}$ of $\mathcal{F}$ and $\mathcal{G}$ of which the domain is $\sim$. By the definition of bisimulation products, the natural projections $f : \mathcal{H} \to \mathcal{F}$ and $g : \mathcal{H} \to \mathcal{G}$ are surjective bounded morphisms. For any proposition letter $p \in \mathsf{props}(\varphi)$, let $V(p) = \{u \mid \mathcal{M}_{T_{\mathcal{ML}}}, f(u) \models p\}$, and for any proposition letter $p \in \mathsf{props}(\psi)$, let $V(p) = \{u \mid \mathcal{N}^+_{T_{\mathcal{ML}}}, g(u) \models p\}$. The properties of $\sim$ guarantee that this V is well-defined for $p \in \mathsf{props}(\varphi) \cap \mathsf{props}(\psi)$. By a standard argument, the graph of f is a bisimulation between $(\mathcal{H}, V)$ and $\mathcal{M}^+_{T_{\mathcal{ML}}}$ with respect to $\mathsf{props}(\varphi)$, and the graph of g is a bisimulation between $(\mathcal{H}, V)$ and $\mathcal{N}^+_{T_{\mathcal{ML}}}$ with respect to $\mathsf{props}(\psi)$.

Now we have the appropriate model in which the contradiction is made explicit, but we have to be able to raise this result to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$. Notice that the model $(\mathcal{H}, V)$ is a tree, since it is the bisimulation product of two trees, and also that the signature of $(\mathcal{H}, V)$ is $\langle \textsc{prop} \cup \{known\}, \textsc{rel}\rangle$. Therefore, we can define the $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-model $(\mathcal{H}, V', S)$ over $\langle \textsc{prop}, \textsc{rel}\rangle$ where $V' = V$ for all $p \in \textsc{prop}$ and $w \in V(known)$ iff $w \in S$. It is easy to see that the equivalent $\mathcal{ML}$-model for $(\mathcal{H}, V', S)$ is $(\mathcal{H}, V)$. So now we need some claim that guarantees us that we can build two relations $\sim_f$ and $\sim_g$ from the graphs of f and g respectively, such that $\sim_f$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $(\mathcal{H}, V', S)$ and $\mathcal{M}^+_T$ and $\sim_g$ is an $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-bisimulation between $(\mathcal{H}, V', S)$ and $\mathcal{N}^+_T$. We will not give the proof of this claim here (refer to [13] for more details). Assuming that we can actually build those relations, it follows that $(\mathcal{H}, V', S), \langle w, v\rangle \models \varphi \wedge \neg\psi$. This contradicts our initial assumption that $\mathcal{C} \models \varphi \to \psi$.

For the $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$ case. Let Tr be the equivalence preserving translation defined in [1] that takes $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$-formulas to $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formulas. Observe that Tr preserves propositional symbols and known, that is, given $\varphi \in \mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$, ⓚ occurs in φ iff ⓚ occurs in $\mathsf{Tr}(\varphi)$ and $\mathsf{props}(\varphi) = \mathsf{props}(\mathsf{Tr}(\varphi))$.

Let φ and ψ be two $\mathcal{ML}^m(\langle\langle r\rangle\rangle, ⓕ)$-formulas such that $\varphi \to \psi$ is valid. Using Tr, we know that $\mathsf{Tr}(\varphi) \to \mathsf{Tr}(\psi)$ is a valid $\mathcal{ML}^m(\langle\langle r\rangle\rangle)$-formula. By Theorem 12, we know that there is an interpolant χ for $\mathsf{Tr}(\varphi)$ and $\mathsf{Tr}(\psi)$ in the common language. Since Tr preserves equivalence, χ is also an interpolant for φ and ψ. Furthermore, given that Tr preserves propositional symbols and known, χ is in the common language of φ and ψ. □

Partial Realization in Dynamic Justification Logic

Samuel Bucheli*, Roman Kuznets**, and Thomas Studer

Institut für Informatik und angewandte Mathematik, Universität Bern,
Bern, Switzerland
{bucheli,kuznets,tstuder}@iam.unibe.ch

Abstract. Justification logic is an epistemic framework that provides a way to express explicit justifications for the agent's belief. In this paper, we present OPAL, a dynamic justification logic that includes term operators to reflect public announcements on the level of justifications. We create dynamic epistemic semantics for OPAL. We also elaborate on the relationship of dynamic justification logics to Gerbrandy–Groeneveld's PAL by providing a partial realization theorem.

1 Introduction

Public announcement logic [14,13] describes how public announcements affect an agent's belief (knowledge). It is a subarea of dynamic epistemic logic [18], which studies the relationship between belief (knowledge) and communication in general. The effect of a public announcement of statement A is represented by a formula $[A]B$ that means *B holds after the public announcement of A*. It is generally assumed that announcements do not affect the material reality of the world, but that the agent trusts the announced facts, partly because their verification may not be an option unless the agent is omniscient. In this paper, we consider beliefs rather than knowledge, hence we concentrate on Gerbrandy–Groeneveld's logic PAL.

The idea of *justification logic* [2] is to formalize reasons for the agent's belief. Instead of $\Box A$ used in the modal language to state that the agent believes A, in justification logic, a specific reason for belief is given—$t : A$ says that *the agent believes A for reason t*. This ability to express explicit reasons for the agent's belief provides a novel approach to several problems of multi-agent systems and formal epistemology. Justifications can be employed, for instance, to tackle the logical omniscience problem [4], to analyze several classical epistemic puzzles [2,3], and to study common knowledge [1,8].

It is natural to ask how public announcements factor into the reasoning of the agent. Public announcement logic describes *how* the beliefs change, but not *why*. The aim of this paper is to suggest ways of formalizing the answer to this *why*. The postulate of Gerbrandy–Groeneveld's PAL that deals with belief change is

$$\Box(A \to [A]B) \leftrightarrow [A]\Box B \ . \tag{1}$$

* Supported by Swiss National Science Foundation grant 200021–117699.

** Supported by Swiss National Science Foundation grant PZ00P2_131706.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 35–51, 2011.

To understand its meaning, it is better to read the equivalence as two separate implications. From left to right, the postulate says that an agent who believes that B must be the case whenever a true fact A is announced will in fact believe B after an actual announcement of A. For instance, elite-level frequent flyers can usually check in for their flight at the business counter by presenting their elite membership card, which can also be attached to their luggage to make public their elite status. This rule should be known to airline employees. The left-to-right implication then means that, when Ann presents her elite membership card to Bob at the business counter, he would know he should check her in. If we are to convert this implication from a statement about beliefs to a statement about reasoning, the result should be something like

$$t:(A \to [A]B) \to [A]s:B \ ,$$

where t represents the airline's regulations regarding business-counter check-in procedures and s is the reason why Bob starts checking Ann in. There are three possibilities how s can be related to t: (A) $s = t$, where the regulations themselves tell Bob to check Ann in; (B) $s = \Uparrow t$, where $\Uparrow$ represents the inference Bob has to make from the regulations after the elite card is shown; and (C) $s = \Uparrow_A t$, where the inference process explicitly mentions both regulations t and the demonstration of Ann's elite card A. In principle, any of the three options can be used to model the situation.

In our joint work with Bryan Renne and Joshua Sack [7], we developed the logic JPAL based on option (A). In this paper, we will present a new logic OPAL based on option (B). We should, therefore, start by explaining why the simplicity of JPAL may not always be sufficient. Imagine that Ann has been upgraded to business class (say, as a reward for postponing her original flight, which was overbooked). So, according to the same regulations, she can check in with Bob based on her ticket alone without announcing her elite status: i.e., $t:B$. But Ann may choose to announce her elite status anyways: i.e., $[A]t:B$. In JPAL, where $s = t$, after the elite status is announced, t encodes two different reasons for Bob to check Ann in. By contrast, in OPAL, these two reasons will be represented by two different terms, t and $\Uparrow t$, of which the latter depends on Ann's elite status while the former is due to the ticket alone. In this situation, Bob would want to distinguish between the two reasons because of the difference in baggage allowances: an elite frequent flyer is often allowed to check more luggage for free than an owner of a business class ticket that has been upgraded from economy.

In addition, in this and similar cases, the JPAL approach implies that the meaning of the regulations changes after public announcements: if Ann has an economy ticket, the regulations do not allow her a business-counter check-in until she shows her elite card, and then they do. This is a little counterintuitive since the regulations are a legal document whose meaning should not be changed by each public announcement. The use of reason $\Uparrow t$ enables us to separate the permanent status of the regulations from their momentary applications influenced by public announcements.

It may seem that adding the exact content of the public announcement to the term $\Uparrow_A t$, as in option (C), provides even more information than the fact of a public announcement in $\Uparrow t$. However, mixing formulas and terms makes the syntax unnecessarily complicated. Worse than that, it makes schematic reasoning, i.e., reasoning invariant with respect to substitutions, impossible. Indeed, the update axiom (1) is actually an axiom scheme that does not depend on the announcement. Therefore, the operation that represents the update on the term level perhaps should not depend on the announcement either. We will, therefore, not consider option (C) in this paper.

One might even consider a fourth option where the reason s is the announcement A itself: i.e., $s = \Uparrow_A$. But the only statement that the announcement of A can plausibly support by itself is A, in which case it should always support it. This would be incompatible with PAL since not all formulas there are successful.

Let us now look at the other implication—from right to left—and see how options (A) and (B) manifest themselves there. The implication states that an agent who will believe B after an announcement of A must believe that, if A is true and announced, B holds after the announcement. For instance, if Charlie, while standing in a long line at the economy check-in counter, sees Ann showing her elite card and being served by Bob at the business counter, $[A]\Box B$, then Charlie has empirical evidence e that Ann is served at the business counter, $[A]e\!:\!B$. It would be natural for Charlie to believe that having an elite status and showing it gets one to the business counter, $\Box(A \to [A]B)$. But it seems even clearer in this case that Charlie's empirical observation e cannot explain the causality of the implication $A \to [A]B$. If before Ann showed up, Charlie had read the sign that invited elite members to the business counter, then Charlie's memory of this sign, refreshed by Ann's actions, could serve as such an explanation. Thus, instead of using e, as in JPAL, in this example it also seems better to use $\Downarrow e$, where $\Downarrow$ is yet another new operation of our logic OPAL.

Apart from the already mentioned [7], not much work has been done so far to combine justification logic and dynamic epistemic logic. A notable exception is Renne's research about introducing new evidence [15] and eliminating unreliable evidence [16] in the framework of justification logic. He also studied the effect certain announcements have on the expressivity of justification logic [17]. However, the modal logic counterparts of those systems do not correspond to any traditional public announcement logic. Both JPAL from [7] and OPAL introduced here are intended as justification logics with public announcement operators whose belief dynamics closely corresponds to the modal belief dynamics of Gerbrandy–Groeneveld's PAL [13].

In this paper, we introduce a justification logic OPAL, *operational public announcement logic*, that extends the language of JPAL with unary operations $\Uparrow$ and $\Downarrow$ to express the occurrence of public announcements and the corresponding change in the justifications of the agent's belief. We present OPAL axiomatically in Sect. 2 and provide it with dynamic epistemic semantics in Sect. 3. After recalling the logic PAL in Sect. 4, in Sect. 5 we prove the soundness and completeness of the newly introduced OPAL with respect to the presented semantics and compare this proof with that in PAL.

While both JPAL and OPAL are intended as exact justification counterparts of PAL, this has to be formally established via the so-called realization theorem. An idea for proving the realization theorem for PAL was sketched in [7], naturally for the logic JPAL discussed there. This idea was dependent on certain properties of JPAL that were conjectured to hold. In Sect. 6 of this paper, we implement this idea and discuss its application to both JPAL and OPAL. The implementation has turned out to be far from being straightforward. In particular, for JPAL, we have established only a partial realization theorem, one that provides a method of substituting terms for $\Box$'s in any theorem of PAL that contains no $\Box$'s within announcements. In contrast to most known constructive proofs of realization theorems, our method, sketched in [7] and implemented here, is not based on a cut-free sequent calculus for PAL. Instead, we first use a modal reduction of PAL to a simpler logic K4, then use the standard realization method for the latter, and finally reverse the performed reduction on the justification side. This reversal process employs the replacement techniques developed by Fitting [11]. The closest analog of this method can be found in [10], where Fitting realizes S5 by reducing it to K45. However, there the reversal of the reduction is trivial. Extending this method to OPAL presented unexpected challenges, which we also discuss at the end of Sect. 6.

Most of the proofs are relegated to the appendix for space considerations.

2 Justification Logic

The language of OPAL extends the language typically used in justification logic by adding public announcement formulas $[A]B$ and two unary operations on terms, $\Uparrow$ and $\Downarrow$, to express the update dynamics of evidence.

Definition 1 (OPAL Language). *We fix countable sets* Cons *of* constants*,* Vars *of* variables*, and* Prop *of* atomic propositions*. The* language of OPAL *consists of the* terms $t \in \mathsf{Tm}$ *and the* formulas $A \in \mathsf{Fml_J}$ *formed by the grammar*

$$t ::= x \mid c \mid (t \cdot t) \mid (t + t) \mid !t \mid \Uparrow t \mid \Downarrow t \qquad x \in \mathsf{Vars},\ c \in \mathsf{Cons}$$

$$A ::= p \mid \neg A \mid (A \to A) \mid t : A \mid [A]A \qquad p \in \mathsf{Prop}$$

Notation 2 (σ-Sequences). *σ and τ (with and without subscripts) will denote finite sequences of formulas. ε denotes the empty sequence. Given such a sequence $\sigma = (A_1, \dots, A_n)$ and a formula B, the formula $[\sigma]B$ is defined as follows:*

$$[\sigma]B := [A_1] \dots [A_n]B \text{ if } n > 0 \qquad \text{and} \qquad [\sigma]B := B \text{ if } n = 0.$$

Further, sequences $\sigma, B := (A_1, \dots, A_n, B)$ and $B, \sigma := (B, A_1, \dots, A_n)$. For another sequence $\tau = (C_1, \dots, C_m)$, we define $\tau, \sigma := (C_1, \dots, C_m, A_1, \dots, A_n)$.

Definition 3 (OPAL Deductive System). *The* axioms of OPAL *consist of all* $\mathsf{Fml_J}$*-instances of the following schemes:*

1. $[\sigma]A$, *where A is a classical propositional tautology*
2. $[\sigma](t : (A \to B) \to (s : A \to t \cdot s : B))$ *(application)*

3. $[\sigma](t:A \to t+s:A), \quad [\sigma](s:A \to t+s:A)$ *(sum)*
4. $[\sigma](t:A \to\, !t:t:A)$ *(introspection)*
5. $[\sigma]p \leftrightarrow p$ *(independence)*
6. $[\sigma](B \to C) \leftrightarrow ([\sigma]B \to [\sigma]C)$ *(normality)*
7. $[\sigma]\neg B \leftrightarrow \neg[\sigma]B$ *(functionality)*
8. $[\sigma]t:(A \to [A]B) \to [\sigma][A]\Uparrow t:B$ *(update $\Uparrow$)*
9. $[\sigma][A]t:B \to [\sigma]\Downarrow t:(A \to [A]B)$ *(update $\Downarrow$)*
10. $[\sigma][A][B]C \leftrightarrow [\sigma][A \wedge [A]B]C$ *(iteration)*

The deductive system OPAL *is a Hilbert system that consists of the above axioms of* OPAL *and the following rules of* modus ponens *(MP) and* axiom necessitation *(AN):*

$$\frac{A \quad A \to B}{B}\,\textit{(MP)}\ , \qquad \frac{c_1,\ldots,c_n \in \mathsf{Cons} \quad C \textit{ is an } \mathsf{OPAL}\textit{-axiom}}{[\sigma_1]c_1 : \cdots : [\sigma_n]c_n : C}\,\textit{(AN)}\ ,$$

where σ_i's are (possibly empty) finite sequences of formulas.

The following example gives some intuition as to how updates are represented by the operations on evidence.

Example 4. For any $p \in \mathsf{Prop}$ and any $c_1, c_2 \in \mathsf{Cons}$, we have $\vdash [p]\Uparrow(c_1 \cdot c_2):p$.

Proof. We use PR to denote the use of propositional reasoning.

1. $c_1:\big(([p]p \leftrightarrow p) \to (p \to [p]p)\big)$	AN for the tautology $([p]p \leftrightarrow p) \to (p \to [p]p)$
2. $c_2:([p]p \leftrightarrow p)$	AN for the independence axiom $[p]p \leftrightarrow p$
3. $(c_1 \cdot c_2):(p \to [p]p)$	from 1 and 2 by application and PR
4. $[p]\Uparrow(c_1 \cdot c_2):p$	from 3 by update $\Uparrow$ and PR □

Note that p need not, in general, be true. The presence of $\Uparrow$ in the term $\Uparrow(c_1 \cdot c_2)$ clearly signifies that this evidence for p is contingent on a prior public announcement. However, the exact content of such a public announcement, p in our case, is not recorded in the term. This design decision enables us to avoid the overcomplexification of the language and is similar to the introspection operation in the traditional justification logics: $!t$ is evidence for $t:A$ whenever t is evidence for A; however, the formula A is not recorded in the term $!t$ either.

Remark 5. The announcement-free fragment of OPAL (the first four axioms with $\sigma = \varepsilon$, rule MP, and rule AN, restricted to $c_n:C$) is the well-known justification logic $\mathsf{J4}$ (see [5]).

We will consider not only OPAL but also JPAL (see [7] for details), which does not include the two term operations $\Uparrow$ and $\Downarrow$.

Definition 6 (JPAL Deductive System). *The* axioms of JPAL *are the axioms of* OPAL *where the two update axiom schemes are replaced by the single scheme*

$$[\sigma]t:(A \to [A]B) \leftrightarrow [\sigma][A]t:B\ . \tag{2}$$

The deductive system JPAL *is a Hilbert system that consists of the axioms of* JPAL *and the rules (MP) and (AN), where the formula C in (AN) now stands for an axiom of* JPAL.

The following lemma states a standard property of justification logics that holds for OPAL and JPAL; it can be proved by an easy induction on the length of derivation.

Lemma 7 (Lifting). *If $s_1 : B_1, \dots, s_m : B_m, C_1, \dots, C_n \vdash A$, then there is a term $t(s_1, \dots, s_m, y_1, \dots, y_n)$ such that*

$$s_1 : B_1, \dots, s_m : B_m, y_1 : C_1, \dots, y_n : C_n \vdash t(s_1, \dots, s_m, y_1, \dots, y_n) : A$$

for fresh variables $y_1, \dots, y_n$.

Corollary 8 (Constructive Necessitation). *For any formula A, if $\vdash A$, then there is a ground term t such that $\vdash t : A$.*

3 Semantics

We adapt the Kripke-style semantics for Justification Logic due to Fitting [9]. A similar semantics for JPAL was presented in [7]. Our semantics uses Kripke models augmented by an *evidence function* that relates each world–term pair (w, t) to a set of formulas $\mathcal{E}(w, t)$ that the term t can justify at the world w.

Definition 9 ($\mathsf{K4}$-Frame). *A* $\mathsf{K4}$-frame *is a pair (W, R) that consists of a set $W \neq \emptyset$ of* (possible) worlds *and of a transitive* accessibility relation $R \subseteq W \times W$.

Definition 10 (Evidence Function). *An* evidence function *on a $\mathsf{K4}$-frame (W, R) is a function $\mathcal{E} \colon W \times \mathsf{Tm} \to \mathcal{P}(\mathsf{Fml_J})$ that satisfies the following closure conditions:*

1. Monotonicity: *if $R(w, v)$, then $\mathcal{E}(w, t) \subseteq \mathcal{E}(v, t)$ for any $t \in \mathsf{Tm}$.*
2. Axioms: *if $c : A$ is derivable by the AN-rule, then $A \in \mathcal{E}(w, c)$ for any $w \in W$.*
3. Application: *if $(A \to B) \in \mathcal{E}(w, t)$ and $A \in \mathcal{E}(w, s)$, then $B \in \mathcal{E}(w, t \cdot s)$.*
4. Sum: $\mathcal{E}(w, s) \cup \mathcal{E}(w, t) \subseteq \mathcal{E}(w, s + t)$ *for any $s, t \in \mathsf{Tm}$ and any $w \in W$.*
5. Introspection: *if $A \in \mathcal{E}(w, t)$, then $t : A \in \mathcal{E}(w, !t)$.*

In a model of OPAL, there is an evidence function $\mathcal{E}^\sigma$ for each finite sequence σ of formulas. The idea is that the evidence function $\mathcal{E}^\sigma$ models the "evidential situation" that arises after the formulas in σ have been publicly announced.

Definition 11 (OPAL Model). *A* model *is a structure $\mathcal{M} = (W, R, \mathcal{E}, \nu)$, where (W, R) is a $\mathsf{K4}$-frame, $\nu : \mathsf{Prop} \to \mathcal{P}(W)$ is a* valuation, *and function $\mathcal{E}$ maps finite sequences σ of formulas to evidence functions $\mathcal{E}^\sigma$ on (W, R) and satisfies*

$$A \to [A]B \in \mathcal{E}^\sigma(w, t) \;\; \textit{implies} \;\; B \in \mathcal{E}^{\sigma, A}(w, \Uparrow t) \ , \tag{3}$$

$$B \in \mathcal{E}^{\sigma, A}(w, t) \;\; \textit{implies} \;\; A \to [A]B \in \mathcal{E}^\sigma(w, \Downarrow t) \ , \tag{4}$$

$$\mathcal{E}^{\sigma, A, B}(w, t) = \mathcal{E}^{\sigma, A \wedge [A]B}(w, t) \ . \tag{5}$$

Conditions (3), (4), and (5) correspond to the update axiom $\Uparrow$, the update axiom $\Downarrow$, and the iteration axiom respectively.

Definition 12 (Truth in OPAL Models). *A ternary relation* $\mathcal{M}, w \Vdash A$ *for* formula A being satisfied at a world $w \in W$ in a model $\mathcal{M} = (W, R, \mathcal{E}, \nu)$ *is defined by induction on the structure of A:*

- $\mathcal{M}, w \Vdash p$ *if and only if* $w \in \nu(p)$.
- *Boolean connectives behave classically.*
- $\mathcal{M}, w \Vdash t\!:\!A$ *if and only if 1)* $A \in \mathcal{E}^{\varepsilon}(w, t)$ *and 2)* $\mathcal{M}, v \Vdash A$ *for all* $v \in W$ *with* $R(w, v)$.
- $\mathcal{M}, w \Vdash [A]B$ *if and only if* $\mathcal{M}_A, w \Vdash B$, *where* $\mathcal{M}_A = (W_A, R_A, \mathcal{E}_A, \nu_A)$ *is defined as follows:* $W_A := W$; $R_A := \{(s,t) \mid R(s,t)$ *and* $\mathcal{M}, t \Vdash A\}$; $(\mathcal{E}_A)^{\sigma} := \mathcal{E}^{A,\sigma}$; *and* $\nu_A := \nu$. *Note that* $\mathcal{M}_A$ *is indeed a model:* R_A *is transitive,* $(\mathcal{E}_A)^{\sigma}$ *is an evidence function on* (W_A, R_A) *for each* σ, *and* $\mathcal{E}_A$ *satisfies conditions* (3)–(5) *from Def. 11.*

We write $\mathcal{M} \Vdash A$ *to mean that* $\mathcal{M}, w \Vdash A$ *for all* $w \in W$. *We say that formula A is* valid, *written* $\Vdash A$, *to mean that* $\mathcal{M} \Vdash A$ *for all models* $\mathcal{M}$. *For a sequence* $\tau = (A_1, \dots, A_n)$ *of formulas we use* $\mathcal{M}_{\tau} = (W_{\tau}, R_{\tau}, \mathcal{E}_{\tau}, \nu_{\tau})$ *to denote the model* $(\cdots((\mathcal{M}_{A_1})_{A_2})\cdots)_{A_n}$. *Note that* $(\mathcal{E}_{\tau})^{\sigma} = \mathcal{E}^{\tau,\sigma}$; *in particular,* $(\mathcal{E}_{\tau})^{\varepsilon} = \mathcal{E}^{\tau}$.

Our notion of model is non-empty as the following example shows.

Example 13. We define the structure $\mathcal{M} = (W, R, \mathcal{E}, \nu)$ as follows: $W := \{w\}$; $R := \{(w,w)\}$; $\mathcal{E}^{\sigma}(w, t) := \mathsf{Fml_J}$ for all σ and all $t \in \mathsf{Tm}$; and $\nu(p) := \{w\}$ for all $p \in \mathsf{Prop}$. It is easy to see that $\mathcal{M}$ is a model.

To illustrate how the semantics works, we prove a semantic version of the result from Example 4.

Example 14. For any $p \in \mathsf{Prop}$ and any $c_1, c_2 \in \mathsf{Cons}$, we have $\Vdash [p] \Uparrow(c_1 \cdot c_2)\!:\!p$.

Proof. Let $\mathcal{M} = (W, R, \mathcal{E}, \nu)$ be an arbitrary model and let $w \in W$. By Def. 10.2, we have $([p]p \leftrightarrow p) \to (p \to [p]p) \in \mathcal{E}^{\varepsilon}(w, c_1)$ and $([p]p \leftrightarrow p) \in \mathcal{E}^{\varepsilon}(w, c_2)$. Thus, $(p \to [p]p) \in \mathcal{E}^{\varepsilon}(w, c_1 \cdot c_2)$ by Def. 10.3. So, by condition (3) from Def. 11, we have $p \in \mathcal{E}^{p}(w, \Uparrow(c_1 \cdot c_2))$. Since $R_p(w, v)$ implies $\mathcal{M}, v \Vdash p$, i.e., $v \in \nu(p) = \nu_p(p)$, we have $\mathcal{M}_p, w \Vdash \Uparrow(c_1 \cdot c_2)\!:\!p$ by Def. 12 and, hence, $\mathcal{M}, w \Vdash [p] \Uparrow(c_1 \cdot c_2)\!:\!p$. □

4 Modal Public Announcement Logic

In this section, we recall some of the basic definitions and facts concerning the Gerbrandy–Groeneveld modal logic of public announcements [12,13,18].

Definition 15 (PAL Language). *The* language of PAL *consists of the* formulas $A \in \mathsf{Fml}_{\Box,[\cdot]}$ *formed by the grammar*

$$A ::= p \mid \neg A \mid (A \to A) \mid \Box A \mid [A]A \qquad p \in \mathsf{Prop}$$

The language $\mathsf{Fml}_{\Box}$ *of modal formulas without announcements is obtained from the same grammar without the* $[A]A$ *constructor.*

The Gerbrandy–Groeneveld theory PAL of Public Announcement Logic uses the language $\mathsf{Fml}_{\Box,[\cdot]}$ to reason about belief change and public announcements.

Definition 16 (PAL Deductive System). *The* axioms of PAL *consist of all* $\mathsf{Fml}_{\Box,[\cdot]}$*-instances of the following schemes:*

1. *Axiom schemes for the modal logic* $\mathsf{K4}$
2. $[A]p \leftrightarrow p$ *(independence)*
3. $[A](B \to C) \leftrightarrow ([A]B \to [A]C)$ *(normality)*
4. $[A]\neg B \leftrightarrow \neg[A]B$ *(functionality)*
5. $[A]\Box B \leftrightarrow \Box(A \to [A]B)$ *(update)*
6. $[A][B]C \leftrightarrow [A \wedge [A]B]C$ *(iteration)*

The deductive system PAL *is a Hilbert system that consists of the above axioms of* PAL *and the following rules of* modus ponens *(MP) and* necessitation *(N):*

$$\frac{A \quad A \to B}{B}\,(MP)\ , \qquad \frac{A}{\Box A}\,(N)\ .$$

We write $\mathsf{PAL} \vdash A$ *to state that* $A \in \mathsf{Fml}_{\Box,[\cdot]}$ *is a theorem of* PAL.

We sometimes use some of the same names for both axioms of OPAL and axioms of PAL because it will always be clear from the context which of the two is meant. As before, the axioms of independence, normality, functionality, update, and iteration are called the *announcement axioms*.

PAL, like many traditional modal public announcement logics, features the so-called *reduction property*: $\mathsf{Fml}_{\Box,[\cdot]}$-formulas with announcements can be reduced to provably equivalent $\mathsf{Fml}_{\Box}$-formulas without announcements [12,13,18]. That means one can express what the situation is after an announcement by saying what the situation was before the announcement. The following lemma formally describes this reduction procedure (for a proof, see, for instance, [18]). This method was first introduced by Plaza in [14].

Definition 17 (Reduction). *The reduction function* $\mathsf{red} : \mathsf{Fml}_{\Box,[\cdot]} \to \mathsf{Fml}_{\Box}$ *is defined as follows:*

1. $\mathsf{red}(p) = p$
2. red *commutes with the connectives* $\neg$, $\to$, *and* $\Box$.
3. $\mathsf{red}([A]p) = p$.
4. $\mathsf{red}([A]\neg B) = \mathsf{red}(\neg[A]B)$.
5. $\mathsf{red}([A](B \to C)) = \mathsf{red}([A]B \to [A]C)$.
6. $\mathsf{red}([A]\Box B) = \mathsf{red}(\Box(A \to [A]B))$.
7. $\mathsf{red}([A][B]C) = \mathsf{red}([A \wedge [A]B]C)$.

Lemma 18 (Provable Equivalence of Reductions). *For all* $A \in \mathsf{Fml}_{\Box,[\cdot]}$, *we have* $\mathsf{PAL} \vdash A \leftrightarrow \mathsf{red}(A)$.

Remark 19. The above lemma facilitates a completeness proof for PAL by reducing it to completeness of $\mathsf{K4}$. Suppose that $A \in \mathsf{Fml}_{\Box,[\cdot]}$ is valid. Then $\mathsf{red}(A)$ is also valid by the soundness of PAL and Lemma 18. Since $\mathsf{red}(A)$ is a formula of $\mathsf{Fml}_{\Box}$, we get by the completeness of $\mathsf{K4}$ that $\mathsf{K4} \vdash \mathsf{red}(A)$ and, hence, that $\mathsf{PAL} \vdash \mathsf{red}(A)$ because PAL extends $\mathsf{K4}$. Applying Lemma 18 again, we conclude that $\mathsf{PAL} \vdash A$.

5 Soundness and Completeness for OPAL

In this section, we establish soundness and completeness of OPAL. First, soundness is shown in the usual way by induction on the length of the derivation.

Lemma 20 (Soundness). *For all formulas A, we have that $\vdash A$ implies $\Vdash A$.*

The traditional modal logic reduction approach (see Remark 19) to establishing completeness is not possible in the presence of justifications since the replacement property does not hold in Justification Logic (see [11, Sect. 6] for a detailed discussion of the replacement property in Justification Logic). That means, in particular, that $\vdash A \leftrightarrow B$ does not imply $\vdash t\!:\!A \leftrightarrow t\!:\!B$, which would be an essential step in the proof of an OPAL-analog of Lemma 18. Thus, it is not possible to transfer the completeness of J4 (see [9]) to OPAL. We will, instead, provide a canonical model construction to prove the completeness of OPAL.

Definition 21 (Maximal Consistent Sets). *A set Φ of $\mathsf{Fml_J}$-formulas is called* consistent *if $\Phi \nvdash A$ for some formula A. A set Φ is called* maximal consistent *if it is consistent but has no consistent proper extensions.*

It can be easily shown that maximal consistent sets contain all axioms of OPAL and are closed under modus ponens and axiom necessitation.

Definition 22 (Canonical Model). *The* canonical model $\mathcal{M} = (W, R, \mathcal{E}, \nu)$ *is defined as follows:*

1. $W := \{w \subseteq \mathsf{Fml_J} \mid w$ *is a maximal consistent set*$\}$,
2. $R(w, v)$ *if and only if for all finite sequences σ and all $t \in \mathsf{Tm}$, we have* $[\sigma]t\!:\!A \in w$ *implies* $[\sigma]A \in v$,
3. $\mathcal{E}^\sigma(w, t) := \{A \in \mathsf{Fml_J} \;:\; [\sigma]t\!:\!A \in w\}$,
4. $\nu(p) := \{w \in W \;:\; p \in w\}$.

Lemma 23 (Truth Lemma). *Let $\mathcal{M}$ be the canonical model. For all formulas D and all worlds w in $\mathcal{M}$, we have $D \in w$ if and only if $\mathcal{M}, w \Vdash D$.*

As usual, the Truth Lemma implies completeness, which, as a corollary, yields announcement necessitation.

Theorem 24 (Completeness). OPAL *is sound and complete: that is, for all formulas $A \in \mathsf{Fml_J}$, we have $\vdash A$ if and only if $\Vdash A$.*

Corollary 25 (Announcement Necessitation). *Announcement necessitation is admissible: that is, for all formulas $A, B \in \mathsf{Fml_J}$, we have $\vdash A$ implies $\vdash [B]A$.*

6 Forgetful Projection and Realization

This section deals with the relationship between PAL and dynamic justification logics. It is easy to show that the forgetful projection A° of an OPAL theorem A is a theorem of PAL. This means that for any theorem A of OPAL, if we replace each term in A with $\Box$, then the resulting formula A° is a theorem of PAL. A similar result holds for JPAL [7].

Definition 26 (Forgetful Projection). *The mapping* $\circ : \mathsf{Fml_J} \to \mathsf{Fml}_{\Box,[\cdot]}$ *is defined as follows:*

$$\begin{array}{ll} p^\circ \quad = p \text{ for all } p \in \mathsf{Prop}, & \circ \text{ commutes with connectives } \neg \text{ and } \to, \\ (t\!:\!A)^\circ = \Box A^\circ & ([A]B)^\circ = [A^\circ]B^\circ \ . \end{array}$$

Theorem 27 (Forgetful Projection of OPAL). *For all formulas* $A \in \mathsf{Fml_J}$*, we have* $\mathsf{OPAL} \vdash A \Longrightarrow \mathsf{PAL} \vdash A^\circ$.

A much more difficult question is whether a dynamic justification logic, such as JPAL or OPAL, can realize PAL: that is, whether for any theorem A of PAL, it is possible to replace each $\Box$ in A with some term such that the resulting formula is a dynamic justification validity.

In the remainder of this paper we present the first realization technique for dynamic justification logics and establish a partial realization result for JPAL: JPAL can realize formulas A that do not contain $\Box$ operators within announcements. Our main idea is to reduce realization of PAL to realization of K4. In our proof, we rely on notions and techniques introduced by Fitting [11].

Definition 28 (Substitution). *A* substitution *is a mapping from variables to terms. If* A *is a formula and* σ *is a substitution, we write* $A\sigma$ *to denote the result of simultaneously replacing each variable* x *in* A *with the term* $x\sigma$.

Lemma 29 (Substitution Lemma). *For every formula* A *of* $\mathsf{Fml_J}$ *and every substitution* σ*, we have* $\vdash A$ *implies* $\vdash A\sigma$.

In most justification logics, in addition to this substitution of proof terms for proof variables, the substitution of formulas for propositions is also possible (see [2]). However, the latter type of substitution typically fails in logics with public announcements, as it does in both JPAL and OPAL.

Definition 30 (Annotations). *An* annotated formula *is a modal formula in which each modal operator is annotated by a natural number. An annotated formula is* properly annotated *if* $\Box_{2k}$*'s occur in it only in negative positions,* $\Box_{2k+1}$*'s occur only in positive positions, and no* $\Box_i$ *occurs twice. Positions within an announcement* $[A]$ *are considered neither positive nor negative: i.e., the parity of indices within announcements in properly annotated formulas is not regulated. If* A' *is the result of replacing all indexed modal operators* $\Box_i$ *with* $\Box$ *in a (properly) annotated formula* A*, then* A *is called a* (properly) annotated version of A'.

Definition 31 (Realization Function). *A* realization function r *is a mapping from natural numbers to terms such that* $r(2i) = x_i$*, where* $x_1, x_2, \ldots$ *is a fixed enumeration of all variables. For a realization function* r *and an annotated formula* A*,* $r(A)$ *denotes the result of replacing each indexed modal operator* $\Box_i$ *in* A *with the term* $r(i)$*. For instance,* $r(\Box_i B) = r(i)\!:\!r(B)$*. A realization function* r *is called* non-self-referential on variables over A *if, for each subformula* $\Box_{2i}B$ *of* A*, the variable* $r(2i) = x_i$ *does not occur in* $r(B)$.

The following realization result for the logic $\mathsf{K4}$ is due to Brezhnev [5]; the additional result about non-self-referentiality can be read off from the proof presented in [6].

Theorem 32 (Realization for $\mathsf{K4}$). *If A is a theorem of $\mathsf{K4}$, then for any properly annotated version B of A, there is a realization function r such that $r(B)$ is provable in $\mathsf{J4}$. Additionally, r is non-self-referential on variables over B.*

In order to formulate the replacement theorem for JPAL, a technical result necessary for demonstrating the partial realization theorem for JPAL, we use the following convention: whenever $D(q)$ and A are formulas in the same language, $D(A)$ is the result of replacing all occurrences of the proposition q in $D(q)$ with A.

For the rest of this section, we consider only formulas A that do not contain modal operators within announcements: i.e., if $[B]C$ is a subformula of A, then B does not contain modal operators.

We will use a theorem that was first formulated by Fitting [11] for LP. A closer look at its proof shows that it also holds in our context. Moreover, while Fitting's formulation only mentions replacement at a positive position, adapting the formulation and the proof to replacement at a negative position is quite easy. The following is an instance of the extended formulation:

Theorem 33 (Replacement for JPAL). *Assume the following:*

1. *$D(q)$ is a properly annotated formula in which the proposition q occurs once and outside of announcements. A and $D(q)$ share no annotations, and B and $D(q)$ share no annotations. If q occurs positively in $D(q)$, then A and B are properly annotated formulas; if q occurs negatively, then $\neg A$ and $\neg B$ are.*
2. *r_1 is a realization function that is non-self-referential on variables over $D(A)$ and $D(B)$.*
3. *If q occurs positively in $D(q)$, then $\mathsf{JPAL} \vdash r_1(A) \to r_1(B)$. If q occurs negatively, then $\mathsf{JPAL} \vdash r_1(B) \to r_1(A)$.*

Then there is some realization/substitution pair $\langle r, \sigma \rangle$ such that

$$\mathsf{JPAL} \vdash r_1(D(A))\sigma \to r(D(B))$$

and r is non-self-referential on variables over $D(B)$.

We now have all ingredients ready to establish our realization theorem. The following diagram shows how we obtain it. We start with a formula $A \in \mathsf{Fml}_{\Box,[\cdot]}$. Using reduction, $\mathsf{K4}$ realization from Theorem 32, and the replacement theorem, we construct a formula $r(A) \in \mathsf{Fml}_{\mathsf{J}}$ that realizes A.

$$\begin{tikzcd}
A \arrow[d, "\text{Reduction}"'] & r(A) \arrow[l, dashed, "\text{Forgetful projection}"'] \\
\mathsf{red}(A) \arrow[r, "\mathsf{K4}\text{ realization}"'] & r_{\mathsf{K4}}(\mathsf{red}(A)) \arrow[u, "\text{Replacement to `invert' }\mathsf{red}"']
\end{tikzcd}$$

Theorem 34 (Realization for PAL). *If A is a theorem of* PAL *such that A does not contain $\Box$ operators within announcements, then for any properly annotated version B of A, there is a realization function r such that $r(B)$ is provable in* JPAL*.*

Remark 35. It is not clear how to generalize our proof to all theorems of PAL. The problem is that the reduction of $[\Box A](C \to D)$ produces two copies of $\Box$ of opposite polarities. Those two occurrences of $\Box$ may then be realized by different terms, and we lack methods of merging terms of opposite polarities in order to 'invert' the reduction.

Remark 36. Unfortunately, adapting this proof to OPAL presents certain challenges. The problem is that in order to 'invert' the reduction from PAL to K4, we need to apply replacement also in negative positions. This is only possible because in the update axiom (2) of JPAL, we have the same justification term on both sides of the equivalence. If, like in OPAL, we work with update operations $\Uparrow$ and $\Downarrow$ on terms, then we end up with different terms in the update axioms, which prevents the use of Fitting's replacement at negative positions.

7 Conclusion

This paper presents OPAL, a dynamic justification logic that includes term operators that reflect public announcements. One of OPAL's update axioms is

$$[\sigma]t:(A \to [A]B) \to [\sigma][A]\Uparrow t:B \;\;,$$

which we used in Example 4 to derive $\vdash [p]\Uparrow(c_1 \cdot c_2):p$. The presence of $\Uparrow$ in the term $\Uparrow(c_1 \cdot c_2)$ clearly points out that this evidence for p is contingent on a prior public announcement.

For the semantics, we employ a combination of epistemic models from justification logic and simple model transformations from dynamic epistemic logics where the agent considers worlds that are inconsistent with the announcement as impossible. We show that OPAL is sound and complete with respect to this semantics.

We develop a realization method for dynamic justification logics and establish a partial realization theorem stating that JPAL realizes all the theorems of PAL that do not contain modalities within announcements. Finally, we discuss why our realization result does not easily transfer to OPAL. It should be noted that our realization method does not rely on cut elimination in PAL, the logic being realized. Its constructiveness, however, depends on cut elimination in K4, to which PAL is reducible.

Acknowledgments. We would like to thank Bryan Renne and Joshua Sack for introducing us to justified dynamic epistemic logic and for many valuable discussions. We thank the anonymous referees for their comments that helped improve the paper. We also thank Galina Savukova for editing this text.

References

1. Artemov, S.N.: Justified common knowledge. Theoretical Computer Science 357(1-3), 4–22 (2006)
2. Artemov, S.N.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
3. Artemov, S.N.: Tracking evidence. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Fields of Logic and Computation. LNCS, vol. 6300, pp. 61–74. Springer, Heidelberg (2010)
4. Artemov, S.N., Kuznets, R.: Logical omniscience as a computational complexity problem. In: Heifetz, A. (ed.) Proc. of TARK 2009, pp. 14–23. ACM, New York (2009)
5. Brezhnev, V.N.: On explicit counterparts of modal logics. Technical Report CFIS 2000–05. Cornell University (2000)
6. Brünnler, K., Goetschi, R., Kuznets, R.: A syntactic realization theorem for justification logics. In: Beklemishev, L., Goranko, V., Shehtman, V. (eds.) Advances in Modal Logic, vol. 8, pp. 39–58. College Publications (2010)
7. Bucheli, S., Kuznets, R., Renne, B., Sack, J., Studer, T.: Justified belief change. In: Arrazola, X., Ponte, M. (eds.) Proc. of the Second ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, pp. 135–155. University of the Basque Country Press (2010)
8. Bucheli, S., Kuznets, R., Studer, T.: Justifications for common knowledge. To appear in the Journal of Applied Non-Classical Logics (2011)
9. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
10. Fitting, M.: The realization theorem for S5, a simple, constructive proof. In: Gupta, A., van Benthem, J. (eds.) Proc. of the Second Indian Conference on Logic and Its Relationship with Other Disciplines (2009) (forthcoming)
11. Fitting, M.: Realizations and LP. Annals of Pure and Applied Logic 161(3), 368–387 (2009)
12. Gerbrandy, J.: Bisimulations on Planet Kripke. PhD thesis. Institute for Logic, Language, and Computation, University of Amsterdam (1999)
13. Gerbrandy, J., Groeneveld, W.: Reasoning about information change. Journal of Logic, Language and Information 6(2), 147–169 (1997)
14. Plaza, J.: Logics of public communications. Synthese 158(2), 165–179 (2007); Reprinted from Emrich, M.L., et al. (eds.) Proc. of ISMIS 1989, pp. 201–216. Oak Ridge National Laboratory, ORNL/DSRD-24 (1989)
15. Renne, B.: Dynamic Epistemic Logic with Justification. PhD thesis. CUNY Graduate Center (2008)
16. Renne, B.: Evidence elimination in multi-agent justification logic. In: Heifetz, A. (ed.) Proc. of TARK 2009, pp. 227–236. ACM, New York (2009)
17. Renne, B.: Public communication in justification logic. Journal of Logic and Computation, Advance Access (July 2010)
18. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. In: Synthese Library, vol. 337. Springer, Heidelberg (2007)

Appendix

Proof of Lemma 20. As usual, the proof is by induction on the length of the derivation of A in OPAL. We only show the most interesting cases for the axioms that relate announcements and justifications.

1. Independence. $\mathcal{M}, w \Vdash [\sigma]p$ iff $\mathcal{M}_\sigma, w \Vdash p$ iff $w \in \nu_\sigma(p)$ iff $w \in \nu(p)$ iff $\mathcal{M}, w \Vdash p$.
2. Update $\Uparrow$. $\mathcal{M}, w \Vdash [\sigma]t : (A \to [A]B)$ is equivalent to the conjunction of
$$A \to [A]B \in \mathcal{E}^\sigma(w, t) \qquad \text{and} \tag{6}$$
$$\mathcal{M}_\sigma, v \Vdash A \to [A]B \text{ for all } v \text{ with } R_\sigma(w, v) \ . \tag{7}$$
By the condition (3) on $\mathcal{E}$ from Def. 11, we obtain that (6) implies
$$B \in \mathcal{E}^{\sigma,A}(w, \Uparrow t) \ . \tag{8}$$
Moreover, (7) is equivalent to
$$\mathcal{M}_\sigma, v \Vdash A \text{ implies } \mathcal{M}_{\sigma,A}, v \Vdash B \text{ for all } v \text{ with } R_\sigma(w, v) \ , \tag{9}$$
which, in turn, is equivalent to
$$\mathcal{M}_{\sigma,A}, v \Vdash B \text{ for all } v \text{ with } R_{\sigma,A}(w, v) \ . \tag{10}$$
The conjunction of this and (8) is equivalent to $\mathcal{M}_{\sigma,A}, w \Vdash \Uparrow t : B$, or, equivalently, $\mathcal{M}, w \Vdash [\sigma][A] \Uparrow t : B$.
3. Update $\Downarrow$. $\mathcal{M}, w \Vdash [\sigma][A]t : B$ is equivalent to the conjunction of (10) and $B \in \mathcal{E}^{\sigma,A}(w, t)$ By the condition (4) on $\mathcal{E}$ from Def. 11, the latter implies
$$A \to [A]B \in \mathcal{E}^\sigma(w, \Downarrow t) \ . \tag{11}$$
Moreover, as noted in the previous case, (10) is equivalent to (7). The conjunction of (7) and (11) is equivalent to $\mathcal{M}_\sigma, w \Vdash \Downarrow t : (A \to [A]B)$, or, equivalently, $\mathcal{M}, w \Vdash [\sigma] \Downarrow t : (A \to [A]B)$.
4. Iteration. First we show that
$$R_{\sigma,A,B} = R_{\sigma,A \wedge [A]B} \ . \tag{12}$$
$R_{\sigma,A,B}(u, v)$ is equivalent to $R_\sigma(u, v)$ and $\mathcal{M}_\sigma, v \Vdash A$ and $\mathcal{M}_{\sigma,A}, v \Vdash B$. This is equivalent to $R_\sigma(u, v)$ and $\mathcal{M}_\sigma, v \Vdash A \wedge [A]B$, which, in turn, is equivalent to $R_{\sigma,A \wedge [A]B}(u, v)$, and thus (12) is established.
The case for iteration is now as follows: $\mathcal{M}, w \Vdash [\sigma][A][B]C$ if and only if $\mathcal{M}_{\sigma,A,B}, w \Vdash C$. By condition (5) on $\mathcal{E}$ from Def. 11 and by (12), this is equivalent to $\mathcal{M}_{\sigma,A \wedge [A]B}, w \Vdash C$, which, in turn, is equivalent to the statement $\mathcal{M}, w \Vdash [\sigma][A \wedge [A]B]C$. □

To establish completeness, we need to know that the canonical model is a model.

Lemma 37 (Correctness of the Canonical Model). *The canonical model is a model.*

Proof. First, we observe that the set W is non-empty: the set of all formulas that are true at world w of the model from Example 13 is maximally consistent. We next show that $\mathcal{E}^\sigma$ is an evidence function on (W, R) for each σ.

- Monotonicity. Assume $A \in \mathcal{E}^\sigma(w,t)$ and $R(w,v)$. We have $[\sigma]t:A \in w$. By introspection and normality, we get $[\sigma]!t:t:A \in w$. By $R(w,v)$, we find $[\sigma]t:A \in v$. Thus, $A \in \mathcal{E}^\sigma(v,t)$.
- Axioms. For any $c:A$ derivable by AN, $[\sigma]c:A$ is also derivable for any σ. Hence, $[\sigma]c:A \in w$ and $A \in \mathcal{E}^\sigma(w,c)$.
- Application. Assume $A \to B \in \mathcal{E}^\sigma(w,t)$ and $A \in \mathcal{E}^\sigma(w,s)$. We then have $[\sigma]t:(A \to B) \in w$ and $[\sigma]s:A \in w$. By application and normality, we get $[\sigma]t \cdot s:B \in w$. Thus, $B \in \mathcal{E}^\sigma(w,t \cdot s)$.
- Sum and Introspection are shown as in the previous case, using the axioms of sum and introspection respectively.

Next we show condition (3) on $\mathcal{E}$ from Def. 11. We have $A \to [A]B \in \mathcal{E}^\sigma(w,t)$ if and only if $[\sigma]t:(A \to [A]B) \in w$. By the update $\Uparrow$ axiom, the latter implies $[\sigma][A]\Uparrow t:B \in w$, which is equivalent to $B \in \mathcal{E}^{\sigma,A}(w,\Uparrow t)$.

Conditions (4) and (5) are shown similarly using the update $\Downarrow$ axiom and the iteration axiom respectively.

Finally, we show that R is transitive. Let $R(w,v)$ and $R(v,u)$. Whenever $[\sigma]t:A \in w$, by introspection and normality, $[\sigma]!t:t:A \in w$. Then $[\sigma]t:A \in v$ since $R(w,v)$. And finally, $[\sigma]A \in u$ since $R(v,u)$. Thus, $R(w,u)$. □

We will need a rank function on formulas to prove the Truth Lemma.

Definition 38 (Rank). *The* rank $\mathsf{rk}(A)$ *of a formula A is defined as follows:*

$$\mathsf{rk}(p) := 1 \textit{ for each } p \in \mathsf{Prop},$$
$$\begin{array}{llll} \mathsf{rk}(\neg A) & := \mathsf{rk}(A)+1\ , & \mathsf{rk}(A \to B) & := \max(\mathsf{rk}(A),\mathsf{rk}(B))+1\ , \\ \mathsf{rk}(t:A) & := \mathsf{rk}(A)+1\ , & \mathsf{rk}([A]B) & := (2+\mathsf{rk}(A)) \cdot \mathsf{rk}(B)\ . \end{array}$$

We immediately get the following lemma:

Lemma 39 (Reductions Reduce Rank). *For all formulas A, B, C and all terms t, we have the following:*

1. $\mathsf{rk}(A) > \mathsf{rk}(B)$ *if B is a proper subformula of A*
2. $\mathsf{rk}([A]\neg B) > \mathsf{rk}(\neg[A]B)$ *3.* $\mathsf{rk}([A](B \to C)) > \mathsf{rk}([A]B \to [A]C)$
4. $\mathsf{rk}([A]t:B) > \mathsf{rk}(\Downarrow t:(A \to [A]B))$ *5.* $\mathsf{rk}([A][B]C) > \mathsf{rk}([A \wedge [A]B]C)$

Proof of Lemma 23. Proof by induction on $\mathsf{rk}(D)$ and a case distinction on the structure of D. Let us only show the cases where D is of the form $[A]B$. The other cases are standard and follow easily from the maximal consistency of w and the definition of the canonical model.

1. $D = [A]p$. Suppose $[A]p \in w$. By the independence axiom, this is equivalent to $p \in w$ for a formula of a lower rank than $[A]p$. By the induction hypothesis, this is equivalent to $\mathcal{M}, w \Vdash p$, which is equivalent to $\mathcal{M}, w \Vdash [A]p$ by the soundness of the independence axiom.
2. $D = [A]\neg B$. Suppose $[A]\neg B \in w$. By the functionality axiom, this is equivalent to $\neg[A]B \in w$ for a formula of a lower rank than $[A]\neg B$. By the induction hypothesis, this is equivalent to $\mathcal{M}, w \Vdash \neg[A]B$, which is equivalent to $\mathcal{M}, w \Vdash [A]\neg B$ by the soundness of the functionality axiom.

3. $D = [A](B \to C)$ is shown similarly using the normality axiom.
4. $D = [A]t\!:\!B$. Suppose $[A]t\!:\!B \in w$. By the update $\Downarrow$ axiom, we then have $\Downarrow t\!:\!(A \to [A]B) \in w$ for a formula of a lower rank than $[A]t\!:\!B$. By the induction hypothesis, this is equivalent to $\mathcal{M}, w \Vdash \Downarrow t\!:\!(A \to [A]B)$, which implies, in particular, that $\mathcal{M}, v \Vdash A \to [A]B$ whenever $R(w, v)$. Equivalently, $\mathcal{M}, v \Vdash [A]B$ whenever $R(w, v)$ and $\mathcal{M}, v \Vdash A$. Equivalently, $\mathcal{M}_A, v \Vdash B$ whenever $R_A(w, v)$. In addition, by the definition of $\mathcal{E}$, we have $B \in \mathcal{E}^A(w, t)$. To summarize, we have $\mathcal{M}_A, w \Vdash t\!:\!B$. In other words, $\mathcal{M}, w \Vdash [A]t\!:\!B$. Suppose $[A]t\!:\!B \notin w$. By the definition of $\mathcal{E}$, we have $B \notin \mathcal{E}^A(w, t)$. Hence, $\mathcal{M}_A, w \nVdash t\!:\!B$. In other words, $\mathcal{M}, w \nVdash [A]t\!:\!B$.
5. $D = [A][B]C$. Suppose $[A][B]C \in w$. By the iteration axiom, this is equivalent to $[A \wedge [A]B]C \in w$ for a formula of a lower rank than $[A][B]C$. By the induction hypothesis, this is equivalent to $\mathcal{M}, w \Vdash [A \wedge [A]B]C$, which, by the soundness of the iteration axiom, is equivalent to $\mathcal{M}, w \Vdash [A][B]C$. □

Proof of Theorem 24. Soundness was already shown in Lemma 20. For completeness, consider the canonical model $\mathcal{M} = (W, R, \mathcal{E}, \nu)$ and assume that $\nvdash A$. Then $\{\neg A\}$ is consistent and, hence, contained in some maximal consistent set $w \in W$. By Lemma 23, it follows that $\mathcal{M}, w \Vdash \neg A$ and, hence, that $\mathcal{M}, w \nVdash A$. Since $\mathcal{M}$ is a model (Lemma 37), we have shown that $\nvdash A$ implies $\nVdash A$. Completeness follows by contraposition. □

Proof of Corollary 25. Assume $\vdash A$. By soundness, $\Vdash A$. Therefore, $\mathcal{M} \Vdash A$ for all models $\mathcal{M}$. In particular, $\mathcal{M}_B, w \Vdash A$ for all models of the form $\mathcal{M}_B$ and worlds w in them. Thus, we obtain $\mathcal{M}, w \Vdash [B]A$ for all $\mathcal{M}, w$. By completeness, we conclude $\vdash [B]A$. □

Proof of Theorem 27. The proof is by induction on the length of the derivation of A in OPAL. For the base case, simply observe that the forgetful projection of each axiom of OPAL is derivable in PAL. The rest is straightforward. □

Proof sketch of Theorem 33. This theorem follows from the JPAL version of [11, Theorem 4.3]. For space considerations we only show the additional case in the algorithm that constructs $\langle r_\varphi, \sigma_\varphi \rangle$ and omit the correctness proof.

Announcement case: $\varphi(P)$ is $[\theta]\eta(P)$, where θ is $\Box_n$-free and $\langle r_\eta, \sigma_\eta \rangle$ has been constructed. Since subformulas of θ are neither positive nor negative, P does not occur in θ. Set $\sigma_\varphi = \sigma_\eta$ and define r_φ as follows:

$$r_\varphi(n) = \begin{cases} r_\eta(n) & \text{if } \Box_n \text{ occurs in } \eta(B) \\ r_1(n) & \text{otherwise.} \end{cases}$$

Proof of Theorem 34. Let A be a formula of $\mathsf{Fml}_{\Box,[\cdot]}$ that does not contain $\Box$ operators within announcements such that $\mathsf{PAL} \vdash A$. By Lemma 18, we find $\mathsf{PAL} \vdash \mathsf{red}(A)$ with no modalities occurring within announcements throughout the reduction process (see Def. 17). By the soundness of PAL, $\mathsf{red}(A)$ is a valid formula of $\mathsf{Fml}_\Box$, and by completeness of $\mathsf{K4}$, we obtain $\mathsf{K4} \vdash \mathsf{red}(A)$.

Let B be a properly annotated version of $\mathsf{red}(A)$. By Theorem 32, there exists a realization function r_{K4} with $\mathsf{J4} \vdash r_{\mathsf{K4}}(B)$ such that r_{K4} is non-self-referential on variables over B. Since $\mathsf{J4}$ is a subsystem of JPAL, we also have $\mathsf{JPAL} \vdash r_{\mathsf{K4}}(B)$.

Now we iteratively 'invert' the reduction steps performed by red. Let us show case 6 in Definition 17. Let $E(q)$ be a formula with one occurrence of q such that $E([C]\Box D)$ has been reduced to the formula $E(\Box(C \to [C]D))$, of which we let $E'(\Box_i(C \to [C]D'))$ be a properly annotated version. Note that C does not contain any modalities and, hence, does not require any annotations. By induction hypothesis, there is a realization function r_1 such that

$$\mathsf{JPAL} \vdash r_1(E'(\Box_i(C \to [C]D'))) \ , \tag{13}$$

which is non-self-referential on that formula. By the JPAL update axiom (2), we have $\mathsf{JPAL} \vdash r_1(\Box_i(C \to [C]D') \leftrightarrow r_1([C]\Box_i D')$. Whether q occurs positively or negatively in $E'(q)$, applying the replacement theorem yields a realization/substitution pair $\langle r, \sigma\rangle$ such that

$$\mathsf{JPAL} \vdash r_1(E'(\Box_i(C \to [C]D')))\sigma \to r(E'([C]\Box_i D')) \ ,$$

with r being non-self-referential on variables over $E'([C]\Box_i D')$. By the substitution lemma, (13) implies $\mathsf{JPAL} \vdash r_1(E'(\Box_i(C \to [C]D')))\sigma$, from which $\mathsf{JPAL} \vdash r(E'([C]\Box_i D'))$ follows by modus ponens.

After having 'inverted' all reduction steps, we end up with a properly annotated version F of A and a realization function r such that $\vdash r(F)$. □

Hoare Logic for Higher Order Store Using Simple Semantics

Nathaniel Charlton

School of Informatics, University of Sussex
n.a.charlton@sussex.ac.uk

Abstract. We revisit the problem of providing a Hoare logic for higher order store programs, considered by Reus and Streicher (ICALP, 2005). In a higher order store program, the procedures/commands of the program are not fixed, but can be manipulated at runtime by the program itself; such programs provide a foundation to study language features such as reflection, dynamic loading and runtime code generation. By adapting the semantics of a proof system for a language with conventional (fixed) mutually recursive procedures, studied by von Oheimb (FSTTCS, 1999), we construct the same logic as Reus and Streicher, but using a much simpler model and avoiding unnecessary restrictions on the use of the proof rules. Furthermore our setup handles nondeterministic programs "for free". We also explain and demonstrate with an example that, contrary to what has been stated in the literature, such a proof system does support proofs which are (in a specific sense) modular.

Keywords: higher order store, Hoare logic, modular proof.

1 Introduction

Higher order store is when a program's commands or procedures are not fixed, but are instead part of the mutable state which the program itself manipulates at runtime. Thus programming languages with higher order store allow programs to be self-modifying and self-configuring. Such languages can be used (as in e.g. [7]) to model phenomena such as runtime code generation and dynamic management of code, which is found in OS kernels, plugin systems and dynamic ("hot") software update systems.

Reus and Streicher [17] consider the problem of providing a Hoare logic for higher order store programs. They consider "arguably the simplest language that uses higher-order store" so that they can focus solely on how to account for higher order store. This language contains programs such as the following:

$$\begin{array}{l} x := \text{‘}\mathsf{run}\ x\text{’}\ ; \\ \mathsf{run}\ x \end{array} \qquad (1) \qquad\qquad \begin{array}{l} x := \text{‘}n := 100\ ;\ x := \text{‘}n := n-1\text{’’}\ ; \\ \mathsf{run}\ x\ ;\ \mathsf{run}\ x \end{array} \qquad (2)$$

Program (1) constructs a non-terminating recursion: the command $\mathsf{run}\ x$ is written into variable x, and then invoked using the $\mathsf{run}\ x$ command. This leads to the code stored in x invoking itself endlessly. Program (1) should satisfy the Hoare

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 52–66, 2011.

triple $\{true\} - \{false\}$. The fact that new recursions can be set up on-the-fly in this way was observed by Landin [9] and is sometimes called *recursion through the store* or *tying the knot*. In program (2) we store a self-modifying command in x: when first run, this command sets n to 100 and then replaces itself with the command $n := n - 1$. Program (2) should satisfy $\{true\} - \{n = 99\}$.

How should one give semantics to such a language? Reus and Streicher take the view that the commands stored in variables should be represented semantically as store transformers, that is, as (partial) functions $\mathsf{Store} \rightharpoonup \mathsf{Store}$. But since stores map variables to values, and values include commands, this makes the notions of command and store mutually recursive. Thus, [17] uses domain theory to solve the following system of recursive equations:

$$\mathsf{Store} = \mathsf{Var} \to \mathsf{Val} \qquad \mathsf{Val} = \mathsf{BVal} + \mathsf{Cmd} \qquad \mathsf{Cmd} = \mathsf{Store} \rightharpoonup \mathsf{Store}$$

where BVal is some basic values such as integers. The rest of the development of *loc. cit.*, which gives an assertion language and Hoare proof rules, then requires a rather intricate domain-theoretic argument. Specifically, results by Pitts [14] concerning the existence of invariant relations on recursively defined domains are needed. Even then, the domain theory leaks out into the logic: several of the rules are restricted in that they can only be used with assertions that are "downwards closed" (a semantic property which may not be intuitive to potential users of the logic, unless they are domain-theorists).

However, when one looks closely at the new proof rules in [17], one sees that they are not so new after all; rather, they are very similar to the rules used by von Oheimb [12] and later others (e.g. Nipkow [11] and Parkinson [13]) for reasoning about mutually recursive procedures in conventional procedural languages (where procedures cannot be updated at runtime). But, noticing this, we are left with an apparent disparity: von Oheimb's proof rules are justified using rather basic mathematics, namely proof by induction on the number of procedure calls made by an execution sequence. If the rules of [17] are so similar, why should the underlying semantics be much more complicated?

In Sections 2, 3 and 4 we resolve this question by showing that using induction on execution length we can reconstruct Reus and Streicher's logic. Not only does this give a simpler model of the logic, but we also eliminate restrictions which [17] places on the use of the proof rules. The key to making this work is to use a simpler, flat model of program states, where all values (including commands) are encoded as integers. In Section 5 we see that our setup handles nondeterministic programs "for free". Section 6 explains that, contrary to what has been stated in the literature, the logic studied here supports proofs which are modular (in a specific sense which we explain). An example of such a proof is provided. Section 7 discusses related and future work, and concludes.

The significance of our results is that they show that, for certain kinds of reasoning about higher order store, it is not necessary to use "sophisticated" domain-theoretic techniques, and in fact one fares better without them.

$$\begin{array}{llll}
\text{expressions} & e & ::= & 0 \mid 1 \mid -1 \mid \ldots \mid e_1 + e_2 \mid e_1 = e_2 \mid \ldots \mid v \mid `C\text{'} \\
\text{commands} & C & ::= & \mathsf{nop} \mid x := e \mid C_1; C_2 \mid \mathsf{if}\ e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2 \mid \mathsf{run}\ x \\
\text{assertions} & P, Q & ::= & P_1 \wedge P_2 \mid \neg P \mid \forall v.P \mid e_1 \leq e_2 \\
\text{contexts} & \Gamma & ::= & \varepsilon \mid \{P\}\, e\, \{Q\}, \Gamma \\
\text{specifications} & S & ::= & \{P\}\, e\, \{Q\} \mid \Gamma \vdash \{P\}\, e\, \{Q\}
\end{array}$$

Fig. 1. Syntax of expressions, commands, assertions, contexts and specifications

$$\frac{}{(x := e, s) \to (\mathsf{nop},\ \lambda v.\ \text{if } v = x \text{ then } [\![e]\!]^{\text{ex}}_s \text{ else } s(v))} \qquad \frac{(C_1, s) \to (C_1', s')}{(C_1; C_2, s) \to (C_1'; C_2, s')}$$

$$\frac{}{(\mathsf{nop}; C, s) \to (C, s)} \qquad \frac{[\![e]\!]^{\text{ex}}_s = 1}{(\mathsf{if}\ e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s) \to (C_1, s)}$$

$$\frac{[\![e]\!]^{\text{ex}}_s \neq 1}{(\mathsf{if}\ e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s) \to (C_2, s)} \qquad \frac{C = \mathscr{G}^{-1}(s(x))}{(\mathsf{run}\ x, s) \to (C, s)}$$

Fig. 2. Small-step operational semantics of programs

2 Programs, Assertions and Specifications

We begin by giving the syntax and semantics of a programming language for higher order store programs, and accompanying assertion and specification languages. The syntax is essentially as in [17] except where indicated, but crucially the semantics takes a different approach.

Let $\mathsf{Store} \triangleq \mathsf{Var} \to \mathbb{Z}$ be the set of program stores and let $\mathsf{Env} \triangleq \mathsf{AuxVar} \to \mathbb{Z}$ be the set of environments for auxiliary variables (throughout we use $\triangleq$ to make definitions, to avoid confusion with the assignment symbol := which occurs in programs). We fix a bijection $\mathscr{G}$ mapping (syntactic) commands to integers; we will use this to encode commands. Thus all values, including commands, are encoded as integers. We call this a *flat* model because, unlike in [17], no complicated order structure on Store is required.

2.1 Expressions, Programs and Assertions

The syntax of expressions e, commands C and assertions P is given in Fig. 1. Variables v can be of two kinds: ordinary variables $x, y, \ldots \in \mathsf{Var}$, and auxiliary variables $\mathrm{p}, \mathrm{q}, \ldots \in \mathsf{AuxVar}$ which may not appear in programs. The quote expression 'C' turns the command C into a value so it can be stored in a variable, and run later. 'C' has no free variables. Using the available assertion language one can express *true*, $\vee$, $\exists$, $=$ and so on in the usual way. Because we use a flat store model and encode commands as integers, we will not need the type check $\tau?e$ from [17], and the typed comparison $\leq_\tau$ becomes $\leq$ on integers.

Semantics of expressions: We write $[\![e]\!]^{\mathrm{ex}}_{s,\rho}$ for the value of expression e in store s and environment ρ. Where e contains no ordinary (resp. auxiliary) variables we omit s (resp. ρ). Expression evaluation is standard apart from the case of 'C', where we simply use the encoding $\mathscr{G}$, defining $[\![\text{`}C\text{'}]\!]^{\mathrm{ex}} \triangleq \mathscr{G}(C)$.

Semantics of programs: Fig. 2 gives a small-step operational semantics[1]. A *configuration* is a pair (C, s) of a command and a store, and is *terminal* if C is nop. One execution step is written $(C, s) \to (C', s')$ and if there is an execution sequence of zero or more steps from (C, s) to (C', s') we write $(C, s) \xrightarrow{*} (C', s')$. Note the semantics of the $\mathsf{run}\ x$ command : we just read an integer value from x, turn it back into a (syntactic) command with $\mathscr{G}^{-1}$, and run it.

Semantics of assertions: The semantics $[\![P]\!]^{\mathrm{as}}_{\rho} \subseteq \mathsf{Store}$ of assertion P in environment ρ is completely standard so we omit it. Entailment $P \Rightarrow Q$ means that for all ρ, $[\![P]\!]^{\mathrm{as}}_{\rho} \subseteq [\![Q]\!]^{\mathrm{as}}_{\rho}$.

2.2 Contexts and Specifications

Next we introduce *contexts* Γ and *specifications* S, also shown in Fig. 1. A context is a collection of Hoare triples. A specification is either a Hoare triple $\{P\}e\{Q\}$, or a triple in context $\Gamma \vdash \{P\}\,e\,\{Q\}$. Intuitively the latter means that $\{P\}\,e\,\{Q\}$ holds in contexts where Γ holds. (In triple $\{P\}\,e\,\{Q\}$ the expression e must not contain free ordinary variables.) Before we can give semantics to contexts and specifications, we state our (partial correctness) interpretation of Hoare triples.

Definition 1. Semantics of Hoare triples. *We write $\rho \models^n \{P\}\,e\,\{Q\}$ to mean, putting $C \triangleq \mathscr{G}^{-1}([\![e]\!]^{\mathrm{ex}}_{\rho})$, that for all stores $s \in [\![P]\!]^{\mathrm{as}}_{\rho}$, if $(C, s) \xrightarrow{*} (\mathsf{nop}, s')$ in n steps or fewer then $s' \in [\![Q]\!]^{\mathrm{as}}_{\rho}$. We write $\rho \models \{P\}\,e\,\{Q\}$ to mean that $\rho \models^n \{P\}\,e\,\{Q\}$ for all $n \in \mathbb{N}$.*

The semantics $[\![S]\!]^{\mathrm{sp}} \subseteq \mathsf{Env}$ of specification S (and the semantics $[\![\Gamma]\!]^{\mathrm{co}} \subseteq \mathsf{Env} \times \mathbb{N}$ of context Γ) is then as in Fig. 3. For example, $\{A\}\,e\,\{B\} \vdash \{P\}\,e'\,\{Q\}$ means that in a context where command e satisfies triple $\{A\}\_\{B\}$ for executions of up to length $n-1$, the command e' satisfies triple $\{P\}\_\{Q\}$ for executions of up to length n. This semantics (which is not a conventional implication) will fit naturally with proof by induction on execution length. Note also that if $(\rho, n) \in [\![\Gamma]\!]^{\mathrm{co}}$ and $0 \leq m < n$ then $(\rho, m) \in [\![\Gamma]\!]^{\mathrm{co}}$.

We write $S_1, \ldots, S_k \models S$ to mean that (the conjunction of) specifications $S_1, \ldots, S_k$ entails the specification S, that is, if $[\![S_1]\!]^{\mathrm{sp}} = \ldots = [\![S_k]\!]^{\mathrm{sp}} = \mathsf{Env}$ then $[\![S]\!]^{\mathrm{sp}} = \mathsf{Env}$. (When $k = 0$ we write just $\models S$ meaning $[\![S]\!]^{\mathrm{sp}} = \mathsf{Env}$.)

3 Proof Rules

In this section we state the proof rules for the logic we work with. The logic features the standard Hoare logic rules for assignment, sequential composition,

[1] We could also carry out our development using denotational semantics, *as long as we kept a flat store model.* The flat store model is the important thing, and we are not trying to say that operational semantics is "better" than denotational semantics.

$$[\![\varepsilon]\!]^{\mathrm{co}} \triangleq \mathsf{Env} \times \mathbb{N} \qquad [\![\{P\}\,e\,\{Q\}\,,\Gamma]\!]^{\mathrm{co}} \triangleq \{(\rho,n) \mid \rho \vDash^{n} \{P\}\,e\,\{Q\}\} \cap [\![\Gamma]\!]^{\mathrm{co}}$$

$$[\![\{P\}\,e\,\{Q\}]\!]^{\mathrm{sp}} \triangleq \{\rho \in \mathsf{Env} \mid \rho \vDash \{P\}\,e\,\{Q\}\}$$

$$[\![\Gamma \vdash \{P\}\,e\,\{Q\}]\!]^{\mathrm{sp}} \triangleq \left\{\rho \in \mathsf{Env} \;\middle|\; \begin{array}{l} \forall n \in \mathbb{N},\ \text{if } n = 0 \text{ or } (n > 0 \text{ and } (\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}) \\ \text{then } \rho \vDash^{n} \{P\}\,e\,\{Q\} \end{array} \right\}$$

Fig. 3. Semantics of contexts Γ and specifications S

$$\text{A}\ \frac{}{\{P[e/x]\}\,\text{`}x := e\text{'}\,\{P\}} \qquad \text{S}\ \frac{\Gamma \vdash \{P\}\,\text{`}C_1\text{'}\,\{R\} \qquad \Gamma \vdash \{R\}\,\text{`}C_2\text{'}\,\{Q\}}{\Gamma \vdash \{P\}\,\text{`}C_1; C_2\text{'}\,\{Q\}}$$

$$\text{I}\ \frac{\Gamma \vdash \{P \wedge e = 1\}\,\text{`}C_1\text{'}\,\{Q\} \qquad \Gamma \vdash \{P \wedge e \neq 1\}\,\text{`}C_2\text{'}\,\{Q\}}{\Gamma \vdash \{P\}\,\text{`}\mathsf{if}\ e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2\text{'}\,\{Q\}}$$

$$\text{W}\ \frac{\Gamma \vdash \{P'\}\,e\,\{Q'\}}{\Gamma \vdash \{P\}\,e\,\{Q\}}\ P \Rightarrow P', Q' \Rightarrow Q \qquad \epsilon\ \frac{}{\{P\}\,\text{`}\mathsf{nop}\text{'}\,\{P\}} \qquad \text{C}\ \frac{\Gamma \vdash \{P\}\,e\,\{Q\}}{T, \Gamma \vdash \{P\}\,e\,\{Q\}}$$

Fig. 4. Standard Hoare logic rules, supported by the logic we study

consequence etc. given in Fig. 4. These are presented as in [17] except that we are explicit about the presence of a context Γ. The interesting rules are those for running stored commands, given in Fig. 5. We can make these rules simpler than those of [17] in two respects, due to our flat store model. Firstly the side conditions about downwards closure of assertions have been eliminated. Secondly we can simply use equality on commands, rather than a partial order $\leq_{\mathrm{com}}$.

Rule (R) is used when we know the stored command C which will be invoked, and have already derived a triple for it. Rule (H) is for making use of contexts: in a context where $\{P \wedge x = \mathrm{p}\}\,\mathrm{p}\,\{Q\}$ holds for executions up to length $n-1$, $\{P \wedge x = \mathrm{p}\}\,\text{`}\mathsf{run}\ x\text{'}\,\{Q\}$ will hold for executions up to length n. Finally rule (μ) is used for reasoning about mutual recursion (through the store): intuitively the premise says that *if* all commands C_j involved satisfy their specifications $\{P_j\}\,\_\,\{Q_j\}$ for executions up to length $n-1$, *then* each command C_i also satisfies its specification for executions up to length n. Unsurprisingly when we later prove soundness of (μ) we will use induction on n.

We will see these rules in action in Section 6. As stated earlier, however, we emphasise that these rules can be seen as adaptations of von Oheimb's rules for (conventional i.e. fixed) recursive procedures to a programming language with higher order store. Specifically, (R) and (μ) taken together strongly resemble the *Call* rule of [12], while (H) strongly resembles the *asm* rule. (For the reader's convenience, the *Call* and *asm* rules are reproduced in the Appendix, Sec. A.2.)

$$\textrm{R}\quad \frac{\Gamma \vdash \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}C\text{’}\, \{Q\}}{\Gamma \vdash \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}\mathsf{run}\ x\text{’}\, \{Q\}} \qquad \textrm{H}\quad \frac{}{\{P \wedge x = \mathrm{p}\}\, \mathrm{p}\, \{Q\} \ \vdash\ \{P \wedge x = \mathrm{p}\}\, \text{‘}\mathsf{run}\ x\text{’}\, \{Q\}}$$

$$\mu\quad \frac{\bigwedge_{1\le i\le N} \ \Gamma, \{P_1\}\, \mathrm{p}_1\, \{Q_1\}, \ldots, \{P_N\}\, \mathrm{p}_N\, \{Q_N\} \ \vdash\ \{P_i\}\, \text{‘}C_i\text{’}\, \{Q_i\}}{\bigwedge_{1\le i\le N} \ \Gamma \vdash \left\{P_i[\vec{C}/\vec{\mathrm{p}}]\right\} \text{‘}C_i\text{’} \left\{Q_i[\vec{C}/\vec{\mathrm{p}}]\right\}} \quad \begin{array}{l} \mathrm{p}_1, \ldots, \mathrm{p}_N \text{ not free in } \Gamma \\ \vec{C} \text{ is } \text{‘}C_1\text{’}, \ldots, \text{‘}C_N\text{’} \\ \vec{p} \text{ is } p_1, \ldots, p_N \end{array}$$

Fig. 5. Hoare logic rules for the run statement which runs stored commands

4 Soundness of the Logic

Having stated the proof rules, we must of course show that they are sound. Soundness proofs for the standard rules (Fig. 4) are straightforward and omitted (except, as an example, the (S) rule is proved in the Appendix, Thm. 3). This leaves the rules for stored commands. We prove soundness of (H) and (μ); as the proofs for (R) and (H) are very similar, we defer the proof of (R) to the Appendix (Thm. 4). As the reader will see, no complicated theory is needed.

Theorem 1. *Rule (H) is sound.*

Proof. Let $\rho \in \mathsf{Env}$, $n \in \mathbb{N}$ be such that $n = 0$ or ($n > 0$ and $(\rho, n-1) \in [\![\{P \wedge x = \mathrm{p}\}\, \mathrm{p}\, \{Q\}]\!]^{\mathrm{co}}$). We must prove $\rho \vDash^n \{P \wedge x = \mathrm{p}\}\, \text{‘}\mathsf{run}\ x\text{’}\, \{Q\}$. If $n = 0$ then this is trivially true, so let $n > 0$. Then from $(\rho, n-1) \in [\![\{P \wedge x = \mathrm{p}\}\, \mathrm{p}\, \{Q\}]\!]^{\mathrm{co}}$ we get (A.) $\rho \vDash^{n-1} \{P \wedge x = \mathrm{p}\}\, \mathrm{p}\, \{Q\}$.

Let $s \in [\![P \wedge x{=}\mathrm{p}]\!]^{\mathrm{as}}_{\rho}$ and s' be such that $(\mathsf{run}\ x, s) \xrightarrow{*} (\mathsf{nop}, s')$ in n steps or fewer; we are required to show $s' \in [\![Q]\!]^{\mathrm{as}}_{\rho}$. Due to the structure of the transition relation $\rightarrow$, we must have $(C, s) \xrightarrow{*} (\mathsf{nop}, s')$ in $n{-}1$ steps or fewer, where $C = \mathscr{G}^{-1}(s(x))$. From this, $s(x) = \rho(\mathrm{p})$ and (A.) we have $s' \in [\![Q]\!]^{\mathrm{as}}_{\rho}$ as required. □

Theorem 2. *Rule (μ) is sound.*

Proof. Let $\Phi(n)$ be the statement that for all $\rho \in \mathsf{Env}$ and all $i \in \{1, \ldots, N\}$,

$$n = 0 \text{ or } (n > 0 \text{ and } (\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}) \quad \text{implies} \quad \rho \vDash^n \{P_i[\vec{C}/\vec{\mathrm{p}}]\}\text{‘}C_i\text{’}\{Q_i[\vec{C}/\vec{\mathrm{p}}]\}$$

It will suffice to prove $\Phi(n)$ for all $n \in \mathbb{N}$, which we shall do by induction.

Base case: Let $\rho \in \mathsf{Env}$ and let $i \in \{1, \ldots, N\}$. Let $\hat{\rho}$ be equal to ρ except at $\mathrm{p}_1, \ldots, \mathrm{p}_N$, which are mapped respectively to $[\![\text{‘}C_1\text{’}]\!]^{\mathrm{ex}}, \ldots, [\![\text{‘}C_N\text{’}]\!]^{\mathrm{ex}}$. From the premise of ($\mu$), unpacking the definitions and instantiating n with 0 and ρ with $\hat{\rho}$, we obtain $\hat{\rho} \vDash^0 \{P_i\}\, \text{‘}C_i\text{’}\, \{Q_i\}$. Using familar properties of substitution, this implies the thing we needed to prove, which is: $\rho \vDash^0 \{P_i[\vec{C}/\vec{\mathrm{p}}]\}\text{‘}C_i\text{’}\{Q_i[\vec{C}/\vec{\mathrm{p}}]\}$.

Inductive case: Let $n > 0$ and let $\Phi(n-1)$ hold. Let $\rho \in \mathsf{Env}$ be such that $(\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$ and let $i \in \{1, \ldots, N\}$. Define $\hat{\rho}$ as in the base case. We must prove $\rho \vDash^n \{P_i[\vec{C}/\vec{\mathrm{p}}]\}\text{‘}C_i\text{’}\{Q_i[\vec{C}/\vec{\mathrm{p}}]\}$ which is equivalent to

$$\hat{\rho} \vDash^n \{P_i\}\text{‘}C_i\text{’}\{Q_i\} \tag{3}$$

using familiar properties of substitution. Note that $(\hat{\rho}, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$ because $(\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$ and $\mathrm{p}_1, \ldots, \mathrm{p}_N$ are not free in Γ. We next show

$$n-1=0 \text{ or } (n-1>0 \text{ and } (\hat{\rho}, n-2) \in [\![\Gamma]\!]^{\mathrm{co}}) \tag{4}$$

Suppose $n-1>0$ i.e. $n>1$. We know that $(\hat{\rho}, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$ and therefore $(\hat{\rho}, n-2) \in [\![\Gamma]\!]^{\mathrm{co}}$. So (4) holds. It now follows from (4) and the induction hypothesis $\Phi(n-1)$, instantiating ρ with $\hat{\rho}$, that

$$\text{for all } j \in \{1, \ldots, N\}, \;\; \hat{\rho} \vDash^{n-1} \{P_j[\vec{C}/\vec{\mathrm{p}}]\}`C_j\text{'}\{Q_j[\vec{C}/\vec{\mathrm{p}}]\}$$

which in turn, using familiar properties of substitution, gives us

$$\text{for all } j \in \{1, \ldots, N\}, \;\; \hat{\rho} \vDash^{n-1} \{P_j\}\mathrm{p}_j\{Q_j\} \tag{5}$$

From the premise of (μ), unpacking the definitions and instantiating n with n and ρ with $\hat{\rho}$, we find that (3) follows from (5) and $(\hat{\rho}, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$. □

5 Nondeterministic Programs

Our setup handles nondeterminism for free, because nowhere in our proofs did we rely on determinism of the transition relation $\rightarrow$. For example, if we add a statement $\mathsf{choose}\ C_1\ C_2$ which makes transitions $(\mathsf{choose}\ C_1\ C_2, s) \rightarrow (C_1, s)$ and $(\mathsf{choose}\ C_1\ C_2, s) \rightarrow (C_2, s)$, it is easy to prove the appropriate Hoare rule:

$$\frac{\Gamma \vdash \{P\}\,`C_1\text{'}\,\{Q\} \qquad \Gamma \vdash \{P\}\,`C_2\text{'}\,\{Q\}}{\Gamma \vdash \{P\}\,`\mathsf{choose}\ C_1\ C_2\text{'}\,\{Q\}}$$

This is in contrast to the approach of [17] where nondeterminism causes problems. This is not an insignificant point: in [16], which extends the ideas of [17] to programs using pointers, a good deal of difficulty is caused by the nondeterminism of dynamic memory allocation. The explanation given is that in the presence of nondeterminism, "programs no longer denote ω-continuous functions".

6 Modular Proofs

We now turn to the issue of modular proofs. In [17] the authors state that their logic "is not modular as all code must be known in advance and must be carried around in assertions"; they further state that it is "highly unlikely" that a modular logic exists at all, ascribing this to the lack of a "Bekic lemma" for their semantics. This belief is reiterated in later work, e.g. in [20]:

> However, the formulation . . . has a shortcoming: code is treated like any other data in that assertions can only mention concrete commands. For modular reasoning, it is clearly desirable to abstract from particular code and instead (partially) specify its behaviour. For example, when verifying mutually recursive procedures on the heap, one would like to consider each procedure in isolation, relying on properties but not the implementations of the others. The recursion rule ... does not achieve this. (6)

(From here on the word "modular" is meant in the sense described in this quote.)

However, it is well known that proof rules such as von Oheimb's support modular proofs; this is the basis of program verifiers (e.g. [10,4]) which check programs one procedure at a time. Therefore, noting their similarity to von Oheimb's rules, it stands to reason that the rules of Reus and Streicher which we work with should already support modular proofs. We now demonstrate using a simple program that this is indeed the case.

Consider the following program C_0:

$$f := `C_1\text{'}\ ;\ g := `C_2\text{'}\ ;\ \mathsf{run}\ f$$

where the commands C_1 and C_2 stored in f and g respectively are defined as follows:

```
C_1 ≜
  if (x × x) + (y × y) = (z × z)
  then nop else run g
```

```
C_2 ≜
  (if x = n then n := n + 1; x := 0
  else if y = n then x := x + 1; y := 0
  else if z = n then y := y + 1; z := 0
  else z := z + 1) ;
  run f
```

This program searches for a Pythagorean triple, that is, numbers x, y, z satisfying the predicate $R(x, y, z) \triangleq x^2 + y^2 = z^2$, stopping when one is found. Note that we establish a mutual recursion *through the store* between the C_1 code (stored in f) and the C_2 code (stored in g). The C_1 code tests whether the current values of variables x, y, z form a Pythagorean triple and terminates if so; otherwise C_1 runs the code in g to continue the search. The C_2 code updates x, y and z to the next triple to try, before invoking the code in f to perform the next test.

We would like to prove that this program works as intended, i.e. satifies

$$\{true\}\, C_0\, \{R(x, y, z)\} \tag{7}$$

But we would also like our proof to be modular, as described in (6), so that we do not have to completely redo our proof if we later change either C_1 (e.g. so that we search for values x, y, z with a different property) or C_2 (e.g. so that we exploit symmetry and only try triples where $x \leq y$). We shall now see how this can be accomplished. Let $T(e)$ be the triple

$$\{\, f = \mathrm{p} \wedge g = \mathrm{q}\,\}\ \ e\ \ \{\, R(x, y, z)\,\}$$

Then, we split our proof into three *independent* pieces.

For C_1: Prove $S_1 \triangleq \quad \vDash T(\mathrm{p}), T(\mathrm{q}) \vdash T(`C_1\text{'})$

For C_2: Prove $S_2 \triangleq \quad \vDash T(\mathrm{p}), T(\mathrm{q}) \vdash T(`C_2\text{'})$

For C_0: Prove $S_0 \triangleq \quad S_1, S_2 \vDash \{true\}\, C_0\, \{R(x, y, z)\}$

Together, these three pieces trivially imply (7). We emphasise that in S_1 above the concrete code for C_2 does *not* appear, only a specification of its behaviour (on the left of $\vdash$), as described in (6). Similarly in S_2 the concrete code for C_1 does not appear, only a specification of its behaviour on the left of $\vdash$.

Proofs of S_0 and S_2 now follow (the proof of S_1 is deferred to the Appendix, Sec. A.1); these proofs demonstrate the use of the (R), (H), and (μ) rules. Note that only the proof for S_1 depends on the definition of predicate R.

Proof (for S_0 piece). In full, the proof obligation S_0 is

$$S_1, S_2 \vDash \{\mathit{true}\}\, f := \text{`}C_1\text{'}\ ;\ g := \text{`}C_2\text{'}\ ;\ \mathsf{run}\ f\, \{R(x, y, z)\}$$

Standard Hoare logic reasoning for assignments and sequential composition reduces this obligation to

$$S_1, S_2 \vDash \{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}\mathsf{run}\ f\text{'}\, \{R(x, y, z)\}$$

By transitivity of $\vDash$ it is enough to show

$$S_1, S_2 \vDash \{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}C_1\text{'}\, \{R(x, y, z)\} \tag{8}$$

and

$$\begin{array}{rl} & \{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}C_1\text{'}\, \{R(x, y, z)\} \\ \vDash & \{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}\mathsf{run}\ f\text{'}\, \{R(x, y, z)\} \end{array} \tag{9}$$

(9) is easily seen to be an instance of rule (R). To deduce (8) we start with the following instance of (μ)

$$\frac{\bigwedge_{i=1,2} \quad T(p), T(q) \ \vdash \ T(\text{`}C_i\text{'})}{\bigwedge_{i=1,2} \ \{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}C_i\text{'}\, \{R(x, y, z)\}}$$

If we use only the $i = 1$ part of the conclusion, we get

$$\frac{T(p), T(q) \vdash T(\text{`}C_1\text{'}) \qquad T(p), T(q) \vdash T(\text{`}C_2\text{'})}{\{f = \text{`}C_1\text{'} \wedge g = \text{`}C_2\text{'}\}\, \text{`}C_1\text{'}\, \{R(x, y, z)\}}$$

which is just (8) written in a different form. □

Proof (for S_2 piece). By the (C) rule, S_2 will follow from $T(\mathrm{p}) \vdash T(\text{`}C_2\text{'})$. By the (S) rule this will follow from

$$T(\mathrm{p}) \vdash \{\, f = \mathrm{p} \wedge g = \mathrm{q}\,\} \begin{array}{l} \text{`}\mathsf{if}\ x = n\ \mathsf{then}\ n := n + 1; x := 0 \\ \mathsf{else\ if}\ y = n\ \mathsf{then}\ x := x + 1; y := 0 \\ \mathsf{else\ if}\ z = n\ \mathsf{then}\ y := y + 1; z := 0 \\ \mathsf{else}\ z := z + 1\text{'} \end{array} \{\, f = \mathrm{p} \wedge g = \mathrm{q}\,\}$$

and

$$T(\mathrm{p}) \vdash \{f = \mathrm{p} \wedge g = \mathrm{q}\}\, \text{`}\mathsf{run}\ f\text{'}\, \{R(x, y, z)\}$$

The former is trivial; the latter is an instance of the (H) rule. □

Suppose we now replace our implementation C_2 with another implementation $\hat{C}_2$, which tries the triples (x, y, z) in a different order. We can reuse our existing proofs of S_1 and S_0; showing $\models T(\mathrm{p}), T(\mathrm{q}) \vdash T(`\hat{C}_2\text{'})$ is the only new work[2,3]. This proof is modular in the same way that proofs about programs with (fixed) recursive procedures can be made modular. If one uses the rules of [12] to show correctness of a program with mutually recursive procedures, and then changes the body of one procedure, then the verification conditions for all other procedures stay the same, and their existing proofs can be reused.

A remark on generalising the (μ) rule

With the (μ) rule we can deal with recursion, even in cases where stored commands participating in a mutual recursion are updated during the recursion. However, as presented here (μ) only allows one to consider *finitely many* commands $C_1, \ldots, C_N$. If one extends the programming language, e.g. with features for runtime code generation, this may no longer be adequate: at runtime a program may have a countably infinite choice of commands it can generate.

We have also developed a generalised version of (μ) which covers these cases. The main idea is that in the generalised rule the variables $\mathrm{p}_1, \ldots, \mathrm{p}_N$ refer not to single commands, but to possibly infinite *sets* of commands. Assertions $x \in \mathrm{p}$ are used instead of $x = \mathrm{p}$, and in the premise, each triple in the context now describes the behaviour of a whole set of commands.

7 Related Work, Future Work and Conclusions

Conclusions

We revisited the Hoare logic given in [17] for a higher order store programming language. We observed that the logic's proof rules strongly resemble those used by von Oheimb [12] for reasoning about programs with conventional (fixed) recursive procedures. This initially appeared puzzling, because whereas von Oheimb's rules are justified using a very simple semantic model, the proofs in [17] depend on the (significantly more complicated) domain-theoretic apparatus developed by Pitts.

We resolved this apparent disparity by giving a simple model of the same logic, using a flat store and induction on execution length to prove the recursion rule (μ). This also allowed us to drop restrictions on the use of the proof rules, and handle nondeterministic programs with no extra work. Finally we demonstrated

[2] Here we see why the lack of a "Bekic lemma" mentioned in [17] is not a problem. When we change the implementation of C_2 to $\hat{C}_2$, from the denotational viewpoint the application of the (μ) rule inside the proof of S_0 just "recomputes" the joint fixed point of C_1 and $\hat{C}_2$.

[3] Strictly, one might wish to explcitly put a universal quantification over R in proof obligations S_2 and S_0. This would be easy to add, but for our present purposes we simply note that the proofs of S_2 and S_0 do not rely on any properties of R, and thus will remain valid proofs whatever the choice of R.

that, contrary to what has been said in [17,20], the logic does support modular proofs.

These results show that, for certain kinds of reasoning about higher order store, it is not necessary to use "sophisticated" domain-theoretic techniques, and in fact one fares better without them.

Related and Future Work

[16,6] study the application of the ideas of [17] to a programming language with a heap, adding *separation logic* connectives [18] to the assertion language. Then, to overcome the perceived lack of modularity of these logics, [20,21] add *nested Hoare triples*. However, these nested triples lead to even greater theoretical complications: [20,21] use Kripke models based on recursively defined ultrametric spaces.

Hence, in the light of what we now know — that the logic of [17] can be given a very simple semantics, and already supports modular proofs — it will be interesting to revisit [20,21] and see whether there is anything which can be accomplished using logics with nested triples, which cannot be supported using a more conventional logic of the kind considered in this paper. The *anti-frame rule* [15,21] may be one such thing, but we cannot yet be sure. On the other hand, the kind of flat model used here can support syntactic operations such as testing syntactic equality of commands, and string-based runtime code generation (described for instance in [1,19] respectively), which it appears the models of [20,21] cannot.

We mention two other approaches to semantically justifying a logic with nested Hoare triples. In [8], total rather than partial correctness is used, and to reason about recursion the user of the logic must essentially perform an induction argument "on foot" in their proofs ([8] was the first paper to give a theory of nested triples, there named *evaluation formulae*). In [5] the *step indexing* technique [2,3] is used, where (unlike here) the interpretation of assertions is also indexed by the length of the execution sequence. Step indexing approaches are very similar in spirit to those based on ultrametric spaces.

Acknowledgements. We thank the anonymous referees for their helpful comments. This research has been supported by EPSRC grant (EP/G003173/1) "*From Reasoning Principles for Function Pointers To Logics for Self-Configuring Programs*".

References

1. Appel, A.W.: Intensional equality ;=) for continuations. SIGPLAN Not. 31, 55–57 (1996)
2. Appel, A.W., McAllester, D.A.: An indexed model of recursive types for foundational proof-carrying code. ACM Trans. Program. Lang. Syst. 23(5), 657–683 (2001)
3. Benton, N., Hur, C.K.: Step-indexing: The good, the bad and the ugly. In: Modelling, Controlling and Reasoning About State. No. 10351 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2010)

4. Berdine, J., Calcagno, C., O'Hearn, P.W.: Smallfoot: Modular automatic assertion checking with separation logic. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 115–137. Springer, Heidelberg (2006)
5. Birkedal, L., Reus, B., Schwinghammer, J., Støvring, K., Thamsborg, J., Yang, H.: Step-indexed Kripke models over recursive worlds. In: POPL, pp. 119–132 (2011)
6. Birkedal, L., Reus, B., Schwinghammer, J., Yang, H.: A simple model of separation logic for higher-order store. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 348–360. Springer, Heidelberg (2008)
7. Charlton, N., Horsfall, B., Reus, B.: Formal reasoning about runtime code update. In: Proceedings of HotSWUp (Hot Topics in Software Upgrades) (to appear, 2011)
8. Honda, K., Yoshida, N., Berger, M.: An observationally complete program logic for imperative higher-order functions. In: LICS, pp. 270–279 (2005)
9. Landin, P.J.: The mechanical evaluation of expressions. Computer Journal 6(4), 308–320 (1964)
10. Møller, A., Schwartzbach, M.I.: The pointer assertion logic engine. In: PLDI 2001 (Programming Language Design and Implementation), pp. 221–231. ACM Press, New York (2001)
11. Nipkow, T.: Hoare logics for recursive procedures and unbounded nondeterminism. In: Bradfield, J.C. (ed.) CSL 2002 and EACSL 2002. LNCS, vol. 2471, pp. 103–119. Springer, Heidelberg (2002)
12. von Oheimb, D.: Hoare logic for mutual recursion and local variables. In: Pandu Rangan, C., Raman, V., Sarukkai, S. (eds.) FST TCS 1999. LNCS, vol. 1738, pp. 168–180. Springer, Heidelberg (1999)
13. Parkinson, M.J.: Local reasoning for Java. Ph.D. thesis, University of Cambridge, Computer Laboratory (November 2005)
14. Pitts, A.M.: Relational properties of domains. Inf. Comput. 127(2), 66–90 (1996)
15. Pottier, F.: Hiding local state in direct style: a higher-order anti-frame rule. In: LICS, Pittsburgh, Pennsylvania, pp. 331–340 (June 2008)
16. Reus, B., Schwinghammer, J.: Separation logic for higher-order store. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 575–590. Springer, Heidelberg (2006)
17. Reus, B., Streicher, T.: About Hoare logics for higher-order store. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1337–1348. Springer, Heidelberg (2005)
18. Reynolds, J.C.: Separation logic: A logic for shared mutable data structures. In: LICS, pp. 55–74 (2002)
19. Richards, G., Lebresne, S., Burg, B., Vitek, J.: An analysis of the dynamic behavior of JavaScript programs. In: PLDI, pp. 1–12 (2010)
20. Schwinghammer, J., Birkedal, L., Reus, B., Yang, H.: Nested hoare triples and frame rules for higher-order store. In: Grädel, E., Kahle, R. (eds.) CSL 2009. LNCS, vol. 5771, pp. 440–454. Springer, Heidelberg (2009)
21. Schwinghammer, J., Yang, H., Birkedal, L., Pottier, F., Reus, B.: A semantic foundation for hidden state. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 2–17. Springer, Heidelberg (2010)

A Appendix

Theorem 3. *Rule (S) is sound.*

$$\text{S}\quad \frac{\Gamma \vdash \{P\}\ `C_1\text{'}\{R\} \qquad \Gamma \vdash \{R\}\ `C_2\text{'}\{Q\}}{\Gamma \vdash \{P\}\ `C_1; C_2\text{'}\{Q\}}$$

Proof. For the sake of clarity, a sound proof rule

$$\frac{S_1 \quad \dots \quad S_k}{S}$$

means that $S_1, \dots, S_k \vDash S$.

Suppose that the premises of (S) hold. We must then show that the conclusion

$$\vDash \Gamma \vdash \{P\}\, `C_1; C_2\text{'}\, \{Q\}$$

holds. So let $\rho \in \mathsf{Env}$ and $n \in \mathbb{N}$ be such that either $n = 0$ or ($n > 0$ and $(\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$). We must show

$$\rho \vDash^n \{P\}\, `C_1; C_2\text{'}\, \{Q\} \tag{10}$$

If $n = 0$ we are done, because $C_1; C_2$ cannot reach a terminal configuration in 0 steps. So assume $n > 0$ and $(\rho, n-1) \in [\![\Gamma]\!]^{\mathrm{co}}$. From this and the premises of (S) we deduce that

$$\rho \vDash^n \{P\}\, `C_1\text{'}\, \{R\} \tag{11}$$

and

$$\rho \vDash^n \{R\}\, `C_2\text{'}\, \{Q\} \tag{12}$$

To prove (10), let

$$(C_1; C_2, s^1) \to \cdots \to (\mathsf{nop}, s^K)$$

be an execution sequence such that $1 < K \leq n + 1$ (so the execution consists of at most n steps) and $s^1 \in [\![P]\!]^{\mathrm{as}}_\rho$. This execution sequence must have the form

$$(C_1; C_2, s^1) \to \cdots \to (\mathsf{nop}; C_2, s^J) \to (C_2, s^{J+1}) \to \cdots \to (\mathsf{nop}, s^K)$$

where $1 \leq J < K$ and $s^J = s^{J+1}$, and there must exist another execution sequence, of $J - 1$ steps, of the form

$$(C_1, s^1) \to \cdots \to (\mathsf{nop}, s^J)$$

By (11) we see that $s^J = s^{J+1} \in [\![R]\!]^{\mathrm{as}}_\rho$. Then applying (12) to the execution sequence

$$(C_2, s^{J+1}) \to \cdots \to (\mathsf{nop}, s^K)$$

we obtain $s^K \in [\![Q]\!]^{\mathrm{as}}_\rho$ as required. □

Theorem 4. *Rule (R) is sound.*

Proof. Suppose that the premise

$$\Gamma \vdash \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}C\text{’}\, \{Q\} \tag{13}$$

holds. We must prove the conclusion

$$\Gamma \vdash \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}\mathsf{run}\ x\text{’}\, \{Q\}$$

So let $\rho \in \mathsf{Env}$, $n \in \mathbb{N}$ be such that $n = 0$ or ($n > 0$ and $(\rho, n-1) \in [\![\varGamma]\!]^{\mathrm{co}}$). We must prove $\rho \vDash^n \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}\mathsf{run}\ x\text{’}\, \{Q\}$. If $n = 0$ then this is trivially true (since $\mathsf{run}\ x$ cannot reach a terminal configuration in 0 steps), so let $n > 0$. The premise (13) gives us

$$(\rho, n-1) \in [\![\varGamma]\!]^{\mathrm{co}} \quad \text{implies} \quad \rho \vDash^n \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}C\text{’}\, \{Q\}$$

But we already know $(\rho, n-1) \in [\![\varGamma]\!]^{\mathrm{co}}$ so we deduce

$$\rho \vDash^n \{P \wedge x = \text{‘}C\text{’}\}\, \text{‘}C\text{’}\, \{Q\} \tag{14}$$

Let $s \in [\![P \wedge x = \text{‘}C\text{’}]\!]^{\mathrm{as}}_{\rho}$ and s' be such that $(\mathsf{run}\ x, s) \xrightarrow{*} (\mathsf{nop}, s')$ in n steps or fewer; we are required to show $s' \in [\![Q]\!]^{\mathrm{as}}_{\rho}$. Due to the structure of the transition relation $\rightarrow$, it must be the case that $(C, s) \xrightarrow{*} (\mathsf{nop}, s')$ in $n-1$ steps or fewer. From this and (14) we have $s' \in [\![Q]\!]^{\mathrm{as}}_{\rho}$ as required. □

A.1 Proof for S_1

Proof (for S_1 piece). By the (C) rule it will suffice to show $T(\mathrm{q}) \vdash T(\text{‘}C_1\text{’})$ i.e.

$$T(\mathrm{q}) \vdash \big\{ f = \mathrm{p} \wedge g = \mathrm{q} \big\} \begin{array}{l} \text{‘}\mathsf{if}\ (x \times x) + (y \times y) = (z \times z) \\ \mathsf{then\ nop} \\ \mathsf{else\ run}\ g\text{’} \end{array} \big\{ R(x,y,z) \big\}$$

Using the (I) rule it will be enough to show

$$T(\mathrm{q}) \vdash \left\{ \begin{array}{l} \quad f = \mathrm{p} \\ \wedge\, g = \mathrm{q} \\ \wedge\, ((x \times x) + (y \times y) = (z \times z)) = 1 \end{array} \right\} \text{‘}\mathsf{nop}\text{’}\ \big\{ R(x,y,z) \big\} \tag{15}$$

and

$$T(\mathrm{q}) \vdash \left\{ \begin{array}{l} \quad f = \mathrm{p} \\ \wedge\, g = \mathrm{q} \\ \wedge\, ((x \times x) + (y \times y) = (z \times z)) \neq 1 \end{array} \right\} \text{‘}\mathsf{run}\ g\text{’}\ \big\{ R(x,y,z) \big\} \tag{16}$$

(15) is easily proved using the definition of $R(x,y,z)$ as $x^2 + y^2 = z^2$ and the equivalence $(e_1{=}e_2) = 1 \Leftrightarrow e_1 = e_2$. We deduce (16) by the (W) rule from the following instance of (H):

$$T(\mathrm{q}) \vdash \{f = \mathrm{p} \wedge g = \mathrm{q}\}\, \text{‘}\mathsf{run}\ g\text{’}\, \{R(x,y,z)\}$$ □

A.2 Two of von Oheimb's Proof Rules

For the reader's convenience we reproduce here the two relevant rules from [12]:

$$\textit{Call}\quad \frac{\Gamma \cup \{\{P_i\}\,\texttt{Call}\;i\,\{Q_i\} \mid i \in ps\} \;\Vdash\; \{\{P_i\}\,\texttt{body}\;i\,\{Q_i\} \mid i \in ps\} \qquad p \in ps}{\Gamma \vdash \{P_p\}\,\texttt{Call}\;p\,\{Q_p\}}$$

$$\textit{asm}\quad \frac{t \in \Gamma}{\Gamma \vdash t}$$

where $\Gamma \vdash t$ abbreviates $\Gamma \Vdash \{t\}$.

The idea of the *Call* rule is that to verify a family *ps* of mutually recursive procedures, one first gives a behavioural specification $\{P_i\}\,\_\,\{Q_i\}$ to each procedure $i \in ps$. One must then prove that the body of each procedure $i \in ps$ actually meets this specification, but when doing so, one is allowed to assume that calls to procedures in *ps* appearing in the body behave as specified. When doing such proofs, the *asm* rule allows one to make use of these assumptions.

Nominal Lawvere Theories

Ranald Clouston[*]

Logic and Computation Group, Research School of Computer Science,
The Australian National University, Canberra, ACT 0200, Australia
ranald.clouston@anu.edu.au

Abstract. Lawvere theories provide a category theoretic view of equational logic, identifying equational theories with small categories equipped with finite products. This formulation allows equational theories to be investigated as first class mathematical entities. However, many formal systems, particularly in computer science, are described by equations modulated by side conditions asserting the "freshness of names"; these may be expressed as theories of Nominal Equational Logic (NEL). This paper develops a correspondence between NEL-theories and certain categories that we call nominal Lawvere theories.

Keywords: Lawvere theory, equational logic, nominal sets, Fraenkel-Mostowski set theory, fresh names.

1 Introduction

Many formal systems, particularly in computer science, may be expressed via equations modulated by side conditions asserting certain names are *fresh for* (not in the free names of) certain meta-variables:

First-order logic: $\Phi \supset (\forall a.\, \Psi) \;=\; \forall a.\, (\Phi \supset \Psi)$ if a is fresh for Φ;
λ-calculus: $\lambda\, a.\, f\, a \;=_\eta\; f$ if a is fresh for f;
π-calculus: $(\nu a\ x) \mid y \;=\; \nu a\, (x \mid y)$ if a is fresh for y.

We may express such modulated equations, and hence reason formally about the systems described by them, with *Nominal Equational Logic (NEL)* [5]. NEL-theories can also express the notions of binding and α-equivalence such systems exhibit [4]. NEL generalises standard equational logic by employing the *nominal sets* [17], and slightly more general *Fraenkel-Mostowski sets (FM-sets)* [9], models, where the manipulation of names is modelled by the action of *permutations.*

Lawvere's view of equational logic [15] identifies *equational theories* with certain *categories*; specifically, small categories with finite products, which are hence called *Lawvere theories.* This category theoretic formulation allows equational theories, as well as their models, to be treated as first class mathematical entities. This has a number of advantages for computer scientists: presentational

[*] The author gratefully acknowledges the supervision of Andrew Pitts and Rajeev Goré, the comments of Richard Garner and the reviewers, and the support of the Woolf Fisher Trust.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 67–83, 2011.

details are abstracted away, theories may be considered in categories other than that of sets, and category theoretic tools, such as sums and monoidal tensor products, may be applied to theories in ways that are more natural than dealing directly with their presentations [11].

The chief development in Lawvere's result is the construction of a *classifying category* for each equational theory.

$$\begin{tikzcd}
\text{equational theories} \arrow[r, bend left, "\text{classifying category}"] & \text{small categories with finite products} \arrow[l, bend left]
\end{tikzcd} \tag{1}$$

In the many-sorted case [6] this category's objects are tuples of sorts $(\mathsf{s}_1, \dots \mathsf{s}_n)$, which correspond to *sorting environments*, arrows are tuples of *terms* modulo provable equivalence, and composition is term *substitution*.

The elegance and usefulness of this perspective has led to a number of papers identifying generalisations of equational logic with the category theoretic structure beyond finite products needed to express them, e.g. [2,20,18]. This paper follows in this tradition by developing Lawvere-style correspondences for NEL.

In expressing theories of NEL as categories, let alone proving a correspondence, we encounter two major hurdles. The first is that variables within terms may have name permutations 'suspended' over them. These *suspensions* are necessary to establish many basic properties, such as $\alpha\beta$-equivalence in the λ-calculus (Ex. 3.3). Suspensions will be captured by a novel structure of arrows in our category, which we call the *internal* Perm-*action* (Def. 4.1).

The second hurdle is that while the objects of Lawvere's classifying categories are sorting environments, NEL has the richer notion of *freshness environments*, where variables are both assigned sorts and may have atoms asserted to be fresh for them. These will be captured by another novel structure called *fresh subobjects* (Def. 4.5).

These concepts, along with *equivariant* finite products (Def. 4.3), define what we will call *FM-categories*, and, where they are small, *nominal Lawvere theories*. An analogue of the correspondence (1) then follows (Sec. 6). The paradigmatic examples of FM-categories will be the category of FM-sets and the classifying category of a NEL-theory. The advantage of this generalised category theoretic view can be most clearly seen in this paper's Completeness Thm. 6.3, whose proof is more conceptually clear and less syntactic than that offered in [5].

The results of this paper are, with some minor differences in presentation, offered in full technical detail in Chap. 7 of the author's thesis [3].

2 Nominal Sets and FM-Sets

Fix a countably infinite set $\mathbb{A}$ of *atoms*, which we will use as names. The set Perm of (finite) *permutations* consists of all bijections $\pi : \mathbb{A} \to \mathbb{A}$ whose domain $supp(\pi) = \{a \mid \pi(a) \neq a\}$ is finite. Perm is generated by *transpositions* $(a\ b)$ that map a to b, b to a and leave all other atoms unchanged. We will make particular use of permutations known as *generalised transpositions* [5, Lem. 10.2]. Let

$$\mathbb{A}^{(n)} \triangleq \{(a_1,\dots,a_n) \in \mathbb{A}^n \mid a_i \neq a_j \text{ for } 1 \leq i < j \leq n\} \tag{2}$$

and take $\vec{a} = (a_1,\dots,a_n), \vec{a}' = (a'_1,\dots,a'_n) \in \mathbb{A}^{(n)}$ with disjoint underlying sets. Then

$$(\vec{a}\,\vec{a}') \triangleq (a_1\; a'_1)\cdots(a_n\; a'_n)$$

Perm may be considered as a one-object category whose arrows are the finite permutations, identity is the trivial permutation ι, and composition is $(\pi' \circ \pi)(a) = \pi'\pi(a) = \pi'(\pi(a))$. A Perm-*set* is then a functor from Perm to the category of sets; that is, a set X equipped with a function, or Perm-action, $(\pi, x) \mapsto \pi \cdot x$ from $\mathsf{Perm} \times X$ to X such that $\iota \cdot x = x$ and $\pi' \cdot (\pi \cdot x) = \pi'\pi \cdot x$.

We say that a set $\overline{a} \subseteq \mathbb{A}$ *supports* $x \in X$ if for all $\pi \in \mathsf{Perm}$, $supp(\pi) \cap \overline{a} = \emptyset$ implies that $\pi \cdot x = x$.

Definition 2.1. *A* nominal set *is a* Perm-*set* X *with the* finite support property*: for each* $x \in X$ *there exists some finite* $\overline{a} \subseteq \mathbb{A}$ *supporting* x*.*

If an element x is finitely supported then there is a unique least such support set [9, Prop. 3.4], which we write $supp(x)$ and call *the support of* x. This may be read as the *set of free names* of a term. If $\overline{a} \cap supp(x) = \emptyset$ for some $\overline{a} \subseteq \mathbb{A}$ we say that $\overline{a}$ *is fresh for* x and write $\overline{a} \# x$, capturing the *not free in* relation.

Example 2.2. (i) Any set becomes a nominal set under the trivial Perm-action $\pi \cdot x = x$, with finite support property $supp(x) = \emptyset$;

(ii) $\mathbb{A}$ is a nominal set with Perm-action $\pi \cdot a = \pi(a)$ and $supp(a) = \{a\}$;

(iii) $\mathbb{A}^{(n)}$ (2), and the set of finite sets of atoms $\mathcal{P}_{fin}(\mathbb{A})$, are nominal sets given the element-wise Perm-actions;

(iv) If X is a nominal set then the *finitely supported powerset* [9, Ex. 3.5]

$$\mathcal{P}_{fs}(X) \triangleq \{S \subseteq X \mid S \text{ is finitely supported in } \mathcal{P}(X)\}$$

is a nominal set given the element-wise Perm-action.

We can define a Perm-action on functions f between Perm-sets by

$$(\pi \cdot f)(x) = \pi \cdot (f(\pi^{-1} \cdot x)) \ . \tag{3}$$

If f has empty support, so that $\pi \cdot (f(x)) = f(\pi \cdot x)$, it is called *equivariant.* The nominal sets and equivariant functions between them form the category $\mathcal{N}om$.

Nominal sets are closed under their Perm-actions, but we can define a more general notion of sets that themselves have finite support. Consider the *Fraenkel-Mostowki hierarchy*:

$$\begin{aligned} \mathcal{FM}_0 &\triangleq \emptyset \\ \mathcal{FM}_{\alpha+1} &\triangleq \mathbb{A} + \mathcal{P}_{fs}(\mathcal{FM}_\alpha) \\ \mathcal{FM}_\lambda &\triangleq \textstyle\bigcup_{\alpha<\lambda} \mathcal{FM}_\alpha \qquad (\lambda \text{ a limit ordinal}) \end{aligned}$$

as α ranges over the ordinals. Each stage defines a nominal set, so the members of the hierarchy are finitely supported (but not necessarily closed) under the evident Perm-action.

Definition 2.3. *The members of the Fraenkel-Mostowki hierarchy are divided by disjoint union into atoms and sets. Call these sets the* FM-sets*.*

Most of the usual set theoretic constructions (with the notable exception of choice) can be performed with FM-sets. In particular, *FM-functions* may be defined, which are finitely supported under the Perm-action ([3]). The FM-sets and FM-functions form the category $\mathcal{FM}$-Set, while small subcategories $\mathcal{FM}_\lambda$-Set may be defined by taking the hierarchy only up to some limit ordinal λ.

3 Nominal Equational Logic

This section summarises Nominal Equational Logic (NEL), mildly generalising [5] so that the collection of sorts forms a nominal set rather a set. This generalisation will be discussed in Rem. 6.6.

Definition 3.1. *A* NEL-signature Σ *is specified by*

(i) *a nominal set* Sort_Σ*, whose elements are called the* sorts of Σ*;*
(ii) *a nominal set* Op_Σ*, whose elements are called the* operation symbols of Σ*;*
(iii) *an equivariant map from each operation symbol* $op \in \mathsf{Op}_\Sigma$ *to a* type*, which we write* $op : (\mathsf{s}_1, \ldots, \mathsf{s}_n) \to \mathsf{s}$*. Where* $n = 0$ *we write* $op : \mathsf{s}$*.*

Fix a countably infinite set Var of *variables.* Then the *terms* over Σ are

$$t ::= \pi\, x \mid op\, t \cdots t$$

for all $\pi \in \mathsf{Perm}, x \in \mathsf{Var}$ and $op \in \mathsf{Op}_\Sigma$. We call $\pi\, x$ a *suspension* and write $\iota\, x$ simply as x. We call $op\, t_1 \cdots t_n$ a *constructed term.*

The *sorting environments* SE_Σ are partial functions $\Gamma : \mathsf{Var} \rightharpoonup \mathsf{Sort}_\Sigma$ with finite domain. We define the set $\Sigma_{\mathsf{s}}(\Gamma)$ of *terms of sort* s *in* Γ by

(i) if $\pi \in \mathsf{Perm}$ and $x \in dom(\Gamma)$ then $\pi\, x \in \Sigma_{\pi \cdot \Gamma(x)}(\Gamma)$;
(ii) given $op : (\mathsf{s}_1, \ldots, \mathsf{s}_n) \to \mathsf{s}$ and $t_i \in \Sigma_{\mathsf{s}_i}(\Gamma)$ for $1 \leq i \leq n$, $op\, t_1 \cdots t_n \in \Sigma_{\mathsf{s}}(\Gamma)$.

The *object-level* Perm-*action* on terms, $(\pi, t \in \Sigma_{\mathsf{s}}(\Gamma)) \mapsto \pi * t \in \Sigma_{\pi \cdot \mathsf{s}}(\Gamma)$, is

$$\begin{aligned} \pi * (\pi'\, x) &\triangleq \pi\pi'\, x \ ; \\ \pi * (op\, t_1 \cdots t_n) &\triangleq (\pi \cdot op)(\pi * t_1) \cdots (\pi * t_n) \ . \end{aligned} \tag{4}$$

This Perm-action is not in general finitely supported, but is used in the definition of substitution: given $\Gamma, \Gamma' \in \mathsf{SE}_\Sigma$, a *substitution* $\sigma : \Gamma \to \Gamma'$ is a map from each $x \in dom(\Gamma)$ to $\sigma(x) \in \Sigma_{\Gamma(x)}(\Gamma')$. Given a term $t \in \Sigma_{\mathsf{s}}(\Gamma)$, the term $t\{\sigma\} \in \Sigma_{\mathsf{s}}(\Gamma')$ is defined by

$$\begin{aligned} (\pi\, x)\{\sigma\} &\triangleq \pi * \sigma(x) \ ; \\ (op\, t_1 \cdots t_n)\{\sigma\} &\triangleq op\, t_1\{\sigma\} \cdots t_n\{\sigma\} \ . \end{aligned} \tag{5}$$

The *freshness environments* FE_Σ are partial functions ∇ with finite domain on Var, mapping each $x \in dom(\nabla)$ to a pair $(\overline{a}, \mathsf{s})$ where $\overline{a} \in \mathcal{P}_{fin}(\mathbb{A})$, $\mathsf{s} \in \mathsf{Sort}_\Sigma$ and $\overline{a} \mathrel{\#} \mathsf{s}$. If $\nabla(x_i) = (\overline{a}_i, \mathsf{s}_i)$ we write ∇ as

$$(\overline{a}_1 \mathrel{\#} x_1 : \mathsf{s}_1, \ldots, \overline{a}_n \mathrel{\#} x_n : \mathsf{s}_n) \ . \tag{6}$$

The intended meaning is that $\overline{a}_i$ is fresh for x_i, which has sort s_i. Each $\nabla \in \mathsf{FE}_\Sigma$ gives rise to a sorting environment $\nabla^{:} \in \mathsf{SE}_\Sigma$ by taking the second projection. We will abbreviate $\{a\} \not\# x : \mathsf{s}$ as $a \not\# x : \mathsf{s}$ and $\emptyset \not\# x : \mathsf{s}$ as $x : \mathsf{s}$. Given a finite set of atoms $\overline{a} \# \nabla$ we can define a new freshness environment by

$$\nabla^{\not\# \overline{a}} \triangleq (\overline{a}_1 \cup \overline{a} \not\# x_1 : \mathsf{s}_1, \ldots, \overline{a}_n \cup \overline{a} \not\# x_n : \mathsf{s}_n) \ . \tag{7}$$

A *NEL-judgement* has the form

$$\nabla \vdash t \approx t' : \mathsf{s} \tag{8}$$

where $\nabla \in \mathsf{FE}_\Sigma$, $\mathsf{s} \in \mathsf{Sort}_\Sigma$ and $t, t' \in \Sigma_\mathsf{s}(\nabla^{:})$. A *NEL-theory* $\mathbb{T}$ is a collection of such judgements.

Fig. 1 present the *proof rules* of NEL, and uses the following new pieces of notation:

- In the rule (SUBST), if ∇ is (6) then $\nabla' \vdash \sigma \approx \sigma'$ stands for the hypotheses $\nabla' \vdash \sigma(x_i) \approx \sigma'(x_i) : \mathsf{s}_i$ for $1 \leq i \leq n$. The notation $\nabla' \vdash \sigma : \nabla$ checks the freshness assumptions of ∇, by $\nabla' \vdash \overline{a}_i \not\# \sigma(x_i) : \mathsf{s}_i$ (see Rem. 3.2).
- $\nabla \leq \nabla'$ if $\nabla(x) = (\mathsf{s}, \overline{a})$ implies $\nabla'(x) = (\mathsf{s}, \overline{a}')$ for $\overline{a} \subseteq \overline{a}'$.
- The *disagreement set* of permutations is $\mathit{ds}(\pi, \pi') \triangleq \{a \in \mathbb{A} \mid \pi(a) \neq \pi'(a)\}$.

If (8) follows from the axioms of $\mathbb{T}$ via these proof rules we write $\nabla \vdash_\mathbb{T} t \approx t' : \mathsf{s}$.

Remark 3.2. In [5] we also used freshness assertions on the right hand side of the turnstile $\vdash$. In fact such judgements do not add expressiveness [3, Sec. 5.5]:

$$\nabla \vdash_\mathbb{T} \overline{a} \not\# t : \mathsf{s} \ \Leftrightarrow\ \nabla^{\not\# \mathit{supp}(\vec{a}')} \vdash_\mathbb{T} t \approx (\vec{a}\ \vec{a}') * t : \mathsf{s} \tag{9}$$

where $\overline{a}$ has ordering $\vec{a} \in \mathbb{A}^{(n)}$ and $\vec{a}' \in \mathbb{A}^{(n)}$ is fresh, so $\mathit{supp}(\vec{a}') \# (\nabla, \overline{a}, t)$. Such freshness judgements will be used only as syntactic sugar in this paper, as equations have a more direct interpretation, as equalities between arrows.

$$(\text{REFL})\ \frac{}{\nabla \vdash t \approx t : \mathsf{s}}\ \nabla \in \mathsf{FE}_\Sigma, t \in \Sigma_\mathsf{s}(\nabla^{:}) \qquad (\text{SYMM})\ \frac{\nabla \vdash t \approx t' : \mathsf{s}}{\nabla \vdash t' \approx t : \mathsf{s}}$$

$$(\text{TRANS})\ \frac{\nabla \vdash t \approx t' : \mathsf{s} \qquad \nabla \vdash t' \approx t'' : \mathsf{s}}{\nabla \vdash t \approx t'' : \mathsf{s}}$$

$$(\text{SUBST})\ \frac{\nabla' \vdash \sigma \approx \sigma' \qquad \nabla' \vdash \sigma : \nabla \qquad \nabla \vdash t \approx t' : \mathsf{s}}{\nabla' \vdash t\{\sigma\} \approx t'\{\sigma'\} : \mathsf{s}}\ \sigma, \sigma' : \nabla^{:} \to (\nabla')^{:}$$

$$(\text{WEAK})\ \frac{\nabla \vdash t \approx t' : \mathsf{s}}{\nabla' \vdash t \approx t' : \mathsf{s}}\ \nabla \leq \nabla' \in \mathsf{FE}_\Sigma \qquad (\text{ATM-ELIM})\ \frac{\nabla^{\not\# \overline{a}} \vdash t \approx t' : \mathsf{s}}{\nabla \vdash t \approx t' : \mathsf{s}}\ \overline{a} \# (\nabla, t, t')$$

$$(\text{PERM})\ \frac{}{\nabla^{\not\# \mathit{ds}(\pi, \pi')} \vdash \pi * t \approx \pi' * t : \pi \cdot \mathsf{s}}\ \nabla \in \mathsf{FE}_\Sigma, t \in \Sigma_\mathsf{s}(\nabla), \mathit{ds}(\pi, \pi') \# (\nabla, t)$$

Fig. 1. Proof rules of Nominal Equational Logic

Example 3.3. A NEL-signature for the untyped λ-calculus can be defined by letting our sorts be the singleton $\{\mathsf{tm}\}$ and operation symbols be

$$\{var_a \mid a \in \mathbb{A}\} \cup \{lam_a \mid a \in \mathbb{A}\} \cup \{app\}$$

with the evident Perm-action (see Ex. 2.2(ii)). The typing function is

$$var_a : \mathsf{tm}, \quad lam_a : (\mathsf{tm}) \to \mathsf{tm}, \quad app : (\mathsf{tm}, \mathsf{tm}) \to \mathsf{tm} \enspace .$$

The binding structure of lam_a is captured, following [4], by the axiom

(α) $(b \,\#\, x : \mathsf{tm}) \vdash lam_a\, x \approx lam_b\, (a\; b)\, x : \mathsf{tm}$

which by (9) may be sugared as $(x : \mathsf{tm}) \vdash a \,\#\, lam_a\, x : \mathsf{tm}$. The NEL-theory for $\alpha\beta\eta$-equivalence, adapting [10, Ex. 2.15], is then (α) plus

(β_1) $(a \,\#\, x : \mathsf{tm}, y : \mathsf{tm}) \vdash app\, (lam_a\, x)\, y \approx x : \mathsf{tm}$
(β_2) $(y : \mathsf{tm}) \vdash app\, (lam_a\, var_a)\, y \approx y : \mathsf{tm}$
(β_3) $(x : \mathsf{tm}, b \,\#\, y : \mathsf{tm}) \vdash app\, (lam_a\, (lam_b\, x))\, y \approx lam_b\, (app\, (lam_a\, x)\, y) : \mathsf{tm}$
(β_4) $(x_1 : \mathsf{tm}, x_2 : \mathsf{tm}, y : \mathsf{tm}) \vdash app\, (lam_a\, (app\, x_1\, x_2))\, y \approx$
$app\, (app\, (lam_a\, x_1)\, y)\, (app\, (lam_a\, x_2)\, y) : \mathsf{tm}$
(β_5) $(b \,\#\, x : \mathsf{tm}) \vdash app\, (lam_a\, x)\, var_b \approx (a\; b)\, x : \mathsf{tm}$
(η) $(a \,\#\, x : \mathsf{tm}) \vdash x \approx lam_a\, (app\, x\, var_a) : \mathsf{tm}$

Example 3.4. For the π-calculus [16] let our sorts be $\{\mathsf{tm}\}$ and our operation symbols include

$$\{res_a \mid a \in \mathbb{A}\} \cup \{in_{a,b} \mid (a, b) \in \mathbb{A} \times \mathbb{A}\} \cup \{par\}$$

representing restriction, input and parallel composition (and in addition to symbols for output and so forth). The evident Perm-actions make these into a nominal set. The typing function is

$$res_a : (\mathsf{tm}) \to \mathsf{tm}, \quad in_{a,b} : (\mathsf{tm}) \to \mathsf{tm}, \quad par : (\mathsf{tm}, \mathsf{tm}) \to \mathsf{tm} \enspace .$$

The axioms for binding are then

(α_1) $(b \,\#\, x : \mathsf{tm}) \vdash res_a\, x \approx res_b\, (a\; b)\, x : \mathsf{tm}$
(α_2) $(c \,\#\, x : \mathsf{tm}) \vdash in_{a,b}\, x \approx in_{a,c}\, (b\; c)\, x : \mathsf{tm}$
(α_3) $(c \,\#\, x : \mathsf{tm}) \vdash in_{a,a}\, x \approx in_{a,c}\, (a\; c)\, x : \mathsf{tm}$

The other axioms for structural congruence are simple to write down, such as

$(-)$ $(x : \mathsf{tm}, a \,\#\, y : \mathsf{tm}) \vdash res_a\, (par\, x\, y) \approx par\, (res_a\, x)\, y$

4 Nominal Lawvere Theories

Definition 4.1. *A category $\mathcal{C}$ has an* internal Perm-action *if for each $\pi \in \mathsf{Perm}$ and $\mathcal{C}$-object C there is a $\mathcal{C}$-arrow π_C with domain C such that*

(i) ι_C *is the identity* id_C*;*
(ii) $(\pi'\pi)_C = \pi'_{\pi\cdot C} \circ \pi_C$*, where* $\pi \cdot C$ *is defined to be the codomain of* π_C*.*

(i) and (ii) specify that the action respects the category structure of Perm. In fact, it is equivalent to a *cofunctor*[1] $\mathsf{Perm} \to \mathcal{C}$, in the sense of [1, Sec. 4.2].

Now given $\pi \in \mathsf{Perm}$ and $\mathcal{C}$-arrow $f : C \to D$, we define the $\mathcal{C}$-arrow $\pi \cdot f : \pi \cdot C \to \pi \cdot D$ to be

$$\begin{tikzcd} \pi \cdot C \arrow[r, "(\pi^{-1})_{\pi\cdot C}"] & C \arrow[r, "f"] & D \arrow[r, "\pi_D"] & \pi \cdot D \end{tikzcd} \quad . \tag{10}$$

The maps $C \mapsto \pi \cdot C$ and $f \mapsto \pi \cdot f$ define an endofunctor $\pi \cdot (\,) : \mathcal{C} \to \mathcal{C}$. In fact, we have a Perm*-category*: a functor from Perm to the category of categories picking out a category $\mathcal{C}$ and defining Perm-actions on its objects $ob\,\mathcal{C}$ and arrows $ar\,\mathcal{C}$. Each π_C is then a component of the natural isomorphism $\pi : id_{\mathcal{C}} \dot{\to} \pi \cdot (\,)$.

Definition 4.2. *An internal* Perm*-action is* finitely supported *if all arrows (and hence objects) are finitely supported under the maps defined above.*

If $\mathcal{C}$ has a finitely supported internal Perm-action and is small, then it is an *internal category in* $\mathcal{N}om$ in the sense of [12, Cha. B2]; its objects and arrows form nominal sets, and the domain, codomain, identity and composition maps are equivariant. Just as standard Lawvere theories have finite products, we will want our category to have finite products in $\mathcal{N}om$:

Definition 4.3. $\mathcal{C}$ *has* equivariant finite products *if it has all finite products so that the terminal object* 1 *has empty support and the maps from tuples of objects* $(C_1, \ldots, C_n)$ *to their projection arrows* $pr_i : C_1 \times \cdots C_n \to C_i$ *are equivariant.*

Example 4.4. Given $\pi \in \mathsf{Perm}$ and FM-set X we define $\pi \cdot X$ by the usual element-wise Perm-action, and the internal Perm-action $\pi_X : X \to \pi \cdot X$ by

$$\pi_X(x) \triangleq \pi \cdot x \ .$$

Defining $\pi \cdot f$ by (10) clearly agrees with (3) and so is finitely supported. $\mathcal{FM}$-Set has finite products defined as usual and it is easy to confirm they are equivariant.

Now given $\overline{a} \in \mathcal{P}_{fin}(\mathbb{A})$ fresh for X, we will define their *fresh subobject* in $\mathcal{FM}$-Set as the inclusion FM-function $i^{\overline{a}}_X : X^{\#\overline{a}} \hookrightarrow X$, where

$$X^{\#\overline{a}} \triangleq \{x \in X \mid \overline{a} \,\#\, x\} \ . \tag{11}$$

We now turn to the arrow-theoretic analogue of (11). The definition below is the most involved of this paper, so first we will provide some motivation. If $\mathcal{P}_{fin}(\mathbb{A})$ is the one object category whose arrows are finite sets of atoms, identity is $\emptyset$ and composition is union, then conditions (i) and (ii) require that fresh subobjects respect that category's structure. $\mathcal{P}_{fin}(\mathbb{A})$ is internal to $\mathcal{N}om$ (though it has no *internal* Perm-action), so (iii) requires that Perm-actions are preserved. (iv) requires that finite products be preserved, while (v)-(vii) are conditions specific to freshness that will be motivated for $\mathcal{FM}$-Set by Ex. 4.6.

[1] The term *cofunctor* is also sometimes used to abbreviate *contravariant functor*.

Definition 4.5. *A category $\mathcal{C}$ with internal* Perm*-action and finite products has* fresh subobjects *if for each finite set of atoms $\overline{a}$ and $\mathcal{C}$-object C such that $\overline{a} \# C$ there is a $\mathcal{C}$-arrow $i_C^{\overline{a}}$ with codomain C so that the following conditions hold:*

(i) $i_C^{\emptyset}$ *is the identity* id_C*;*

(ii) $i_C^{\overline{a} \cup \overline{a}'} = i_C^{\overline{a}} \circ i_{C^{\not\#\overline{a}}}^{\overline{a}'}$*, where $C^{\not\#\overline{a}}$ is defined to be the domain of $i_C^{\overline{a}}$.*

(iii) $\pi \cdot i_C^{\overline{a}} = i_{\pi \cdot C}^{\pi \cdot \overline{a}}$.

(iv) $i_{C_1 \times \cdots \times C_n}^{\overline{a}} = i_{C_1}^{\overline{a}} \times \cdots \times i_{C_n}^{\overline{a}}$.

(v) (Fresh permutations): Given $\pi \in \mathsf{Perm}$*, if $supp(\pi) \# C$ then $\pi_{C^{\not\# supp(\pi)}}$ is the identity $id_{C^{\not\# supp(\pi)}}$;*

(vi) (Fresh epis): If we have a finite set of atoms $\overline{a}$ and parallel $\mathcal{C}$-arrows $f : C \rightrightarrows D$ such that $f \circ i_C^{\overline{a}} = g \circ i_C^{\overline{a}}$ and $\overline{a} \# (f, g)$, then $f = g$;

(vii) (Fresh arrows): Suppose we have sets of atoms $\overline{a}, \overline{a}'$ which may be ordered $\vec{a}, \vec{a}' \in \mathbb{A}^{(n)}$, and a $\mathcal{C}$-arrow $f : D \to C$. If $\overline{a} \# C$, $\overline{a}' \# (\overline{a}, f)$ and $(\vec{a}\;\vec{a}')_C \circ f \circ i_D^{\overline{a}'} = f \circ i_D^{\overline{a}'}$, then we have a unique $\hat{f} : D \to C^{\not\#\overline{a}}$ such that $i_C^{\overline{a}} \circ \hat{f} = f$:

$$\begin{tikzcd}
D^{\not\#\overline{a}'} \arrow[r, "i_D^{\overline{a}'}"] & D \arrow[d, dotted, "\hat{f}"'] \arrow[dr, "f"] & \\
& C^{\not\#\overline{a}} \arrow[r, "i_C^{\overline{a}}"'] & C \arrow[loop right, "(\vec{a}\;\vec{a}')_C"]
\end{tikzcd} \tag{12}$$

Example 4.6. Conditions (v)-(vii) above applied to (11) in $\mathcal{FM}$-Set ask that

- If we have $x \in X$ and $supp(\pi) \# (x, X)$ then $\pi \cdot x = x$;
- If $\overline{a} \# (f, g)$, and $f(x) = g(x)$ whenever $\overline{a} \# x$, then $f = g$;
- Say $\overline{a} \# C$ and $\overline{a}' \# (\overline{a}, f)$. If $(\vec{a}\;\vec{a}') \cdot f(x) = f(x)$ whenever $\overline{a}' \# x$, then $\overline{a} \# f(x)$ for all x.

These are all readily verifiable properties of FM-sets and functions. In particular, (vii) encodes the freshness property (9) for arrows of our category. $\mathcal{FM}$-Set and the small $\mathcal{FM}_\lambda$-Set therefore fulfil the conditions of the following definition.

Definition 4.7. *An* FM-category *is a category with a finitely supported internal* Perm*-action, equivariant finite products and fresh subobjects. A small FM-category will be called a* nominal Lawvere theory*.*

From now on we will abbreviate the internal Perm-action π_C to π and fresh subobjects $i_C^{\overline{a}}$ to $i^{\overline{a}}$ where this is clear. The following definition and lemma demonstrate the utility of the Fresh Arrows condition above.

Definition 4.8. *Suppose that $\overline{a} \# (f : C \to D)$ in $\mathcal{C}$. Let $\vec{a} \in \mathbb{A}^{(n)}$ be an ordering of $\overline{a}$ and $\vec{a}' \in \mathbb{A}^{(n)}$ be a fresh tuple of the same size. $(\vec{a}\;\vec{a}')$ defines a natural transformation $id_{\mathcal{C}} \dot{\to} (\vec{a}\;\vec{a}') \cdot ()$, so*

$$\begin{tikzcd}
C^{\not\#\,\overline{a} \cup supp(\vec{a}')} \arrow[r, "i^{\not\#\,\overline{a} \cup supp(\vec{a}')}"] \arrow[d, "(\vec{a}\;\vec{a}')"'] & C \arrow[r, "f"] & D \arrow[d, "(\vec{a}\;\vec{a}')"] \\
C^{\not\#\,\overline{a} \cup supp(\vec{a}')} \arrow[r, "i^{\not\#\,\overline{a} \cup supp(\vec{a}')}"'] & C \arrow[r, "f"'] & D
\end{tikzcd}$$

commutes. But $(\vec{a}\ \vec{a}')_{C^{\#\overline{a}\cup supp(\vec{a}')}}$ *is the identity by Def. 4.5(v), so* $(\vec{a}\ \vec{a}') \circ f \circ i^{\#\overline{a}\cup supp(\vec{a}')} = f \circ i^{\#\overline{a}\cup supp(\vec{a}')}$. *Therefore by Def. 4.5(vii), the Fresh Arrows condition, we induce a unique arrow which we will call* $f^{\#\overline{a}}$*:*

$$\begin{tikzcd}
C^{\#\overline{a}\cup supp(\vec{a}')} \arrow[r, "i^{supp(\vec{a}')}"] & C^{\#\overline{a}} \arrow[r, "i^{\overline{a}}"] \arrow[d, dotted, "f^{\#\overline{a}}"'] & C \arrow[d, "f"] \\
 & D^{\#\overline{a}} \arrow[r, "i^{\overline{a}}"'] & D \arrow[loop right, "(\vec{a}\ \vec{a}')"]
\end{tikzcd}$$

Lemma 4.9. *All fresh subobjects* $i^{\overline{a}}_C : C^{\#\overline{a}} \to C$ *are mono.*

Proof. Say $i^{\overline{a}} \circ f = i^{\overline{a}} \circ g$ for $f, g : D \rightrightarrows C^{\#\overline{a}}$, and take $\vec{a}$ as an ordering of $\overline{a}$ and $\vec{a}'$ as a fresh tuple of the same size. $(\vec{a}\ \vec{a}')_C \circ i^{\overline{a}}_C \circ f \circ i^{supp(\vec{a}')}_D = (\vec{a}\ \vec{a}')_C \circ i^{\overline{a}\cup supp(\vec{a}')}_C \circ f^{\# supp(\vec{a}')}$ by Def. 4.8. By the naturality of $(\vec{a}\ \vec{a}')$ and Def 4.5(v) this equals $i^{\overline{a}}_C \circ f \circ i^{supp(\vec{a}')}_D$. Therefore by Def. 4.5(vii) we have a unique arrow:

$$\begin{tikzcd}
D^{\# supp(\vec{a}')} \arrow[r, "i^{supp(\vec{a}')}"] & D \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, dotted] & C^{\#\overline{a}} \arrow[d, "i^{\overline{a}}"] \\
 & C^{\#\overline{a}} \arrow[r, "i^{\overline{a}}"'] & C \arrow[loop right, "(\vec{a}\ \vec{a}')"]
\end{tikzcd}$$

But $i^{\overline{a}} \circ f = i^{\overline{a}} \circ g$, so by uniqueness $f = g$.

Definition 4.10. *Given FM-categories* $\mathcal{C}, \mathcal{C}'$, *an* FM-functor $\mathcal{C} \to \mathcal{C}'$ *is a functor that strictly preserves*

(i) the internal Perm*-action:* $F(\pi_C) = \pi_{FC}$*;*
(ii) finite products: $F(pr_i) = pr_i : FC_1 \times \cdots \times FC_n \to FC_i$*;*
(iii) fresh subobjects: $F(i^{\overline{a}}_C) = i^{\overline{a}}_{FC}$*.*

$FM(\mathcal{C}, \mathcal{C}')$ *is the category of FM-functors* $\mathcal{C} \to \mathcal{C}'$ *and natural transformations.*

5 Algebra in FM-Categories

Definition 5.1. *Given a NEL-signature* Σ *and FM-category* $\mathcal{C}$, *a* Σ-structure M *in* $\mathcal{C}$ *is defined by*

(i) An equivariant function $M[\![-]\!] : \mathsf{Sort}_\Sigma \to ob\,\mathcal{C}$*;*
(ii) An equivariant function $M[\![-]\!] : \mathsf{Op}_\Sigma \to ar\,\mathcal{C}$ *where*

$$M[\![op]\!] : M[\![\mathsf{s}_1]\!] \times \cdots \times M[\![\mathsf{s}_n]\!] \to M[\![\mathsf{s}]\!] \tag{13}$$

if op has type $(\mathsf{s}_1, \ldots, \mathsf{s}_n) \to \mathsf{s}$.

Where the structure in question is clear we will write $M[\![\mathsf{s}]\!]$ *as* $[\![\mathsf{s}]\!]$ *and so forth.*

Definition 5.2. *Given a freshness environment ∇ as (6) and a Σ-structure in an FM-category $\mathcal{C}$ we define the $\mathcal{C}$-object*

$$\llbracket\nabla\rrbracket \;\triangleq\; \llbracket\mathsf{s}_1\rrbracket^{\#\overline{a}_1} \times \cdots \times \llbracket\mathsf{s}_n\rrbracket^{\#\overline{a}_n} \;\; .$$

Given a term $t \in \Sigma_{\mathsf{s}}(\nabla^{\cdot})$ the value arrow $\llbracket\nabla \vdash t : \mathsf{s}\rrbracket$ *is a $\mathcal{C}$-arrow $\llbracket\nabla\rrbracket \to \llbracket\mathsf{s}\rrbracket$:*

$$\begin{aligned}\llbracket\nabla \vdash \pi\, x_i : \mathsf{s}_i\rrbracket \;&\triangleq\; \pi_{\llbracket\mathsf{s}_i\rrbracket} \circ i^{\overline{a}_i}_{\llbracket\mathsf{s}_i\rrbracket} \circ pr_i \;\; ; \\ \llbracket\nabla \vdash op\, t_1 \cdots t_n : \mathsf{s}\rrbracket \;&\triangleq\; \llbracket op\rrbracket \circ \langle \llbracket\nabla \vdash t_1 : \mathsf{s}_1\rrbracket, \ldots, \llbracket\nabla \vdash t_n : \mathsf{s}_n\rrbracket\rangle \;\; .\end{aligned}$$

Definition 5.3. *A structure M in an FM-category $\mathcal{C}$* satisfies *the judgement $\nabla \vdash t \approx t' : \mathsf{s}$ if $M\llbracket\nabla \vdash t : \mathsf{s}\rrbracket = M\llbracket\nabla \vdash t' : \mathsf{s}\rrbracket$. If M satisfies all axioms of a theory $\mathbb{T}$ then it is a* $\mathbb{T}$-algebra in $\mathcal{C}$*.*

Theorem 5.4 (Soundness). *If M is a $\mathbb{T}$-algebra in $\mathcal{C}$ and $\nabla \vdash_{\mathbb{T}} t \approx t' : \mathsf{s}$ then M satisfies that judgement.*

Proof. We need to show closure under the proof rules of Fig. 1; see App. A.1.

Definition 5.5. *A* $\mathbb{T}$-homomorphism $M \to M'$ *in $\mathcal{C}$ is an equivariant function h from Sort_{Σ} to $\mathcal{C}$-arrows, with $h(\mathsf{s}) = h_{\mathsf{s}} : M\llbracket\mathsf{s}\rrbracket \to M'\llbracket\mathsf{s}\rrbracket$, such that*

$$h_{\mathsf{s}} \circ M\llbracket op\rrbracket = M'\llbracket op\rrbracket \circ (h_{\mathsf{s}_1} \times \cdots \times h_{\mathsf{s}_n}) \tag{14}$$

for all op. The $\mathbb{T}$-algebras in $\mathcal{C}$ and $\mathbb{T}$-homomorphisms form the category $\mathcal{C}^{\mathbb{T}}$.

6 Category-Theory Correspondence

Given a NEL-theory $\mathbb{T}$, this section defines a nominal Lawvere theory called the *classifying category*, $Cl(\mathbb{T})$. This construction gives rise to a simple completeness proof, along with the key correspondences of this paper.

Lemma 6.1. *Fix an ordering $v_1, v_2, \ldots$ on the set of variables Var and let $\mathbb{T}$ be a theory over a signature Σ. Then the following constructions define a nominal Lawvere theory, which we will call the* classifying category *and write $Cl(\mathbb{T})$.*

Objects: *$ob\, Cl(\mathbb{T})$ is the set of freshness environments whose domain is an initial sublist of Var, $\{v_1, \ldots, v_n\}$, so the typical object is*

$$\nabla \;=\; (\overline{a}_1 \# v_1 : \mathsf{s}_1, \ldots, \overline{a}_n \# v_n : \mathsf{s}_n) \tag{15}$$

Arrows: *Taking ∇ as (15), $Cl(\mathbb{T})$-arrows $f : \nabla' \to \nabla$ are defined by*

$$\nabla' \vdash (\overline{a}_1 \# [t_1] : \mathsf{s}_1, \ldots, \overline{a}_n \# [t_n] : \mathsf{s}_n) \tag{16}$$

where, for $1 \leq i \leq n$,
(i) $t_i \in \Sigma_{\mathsf{s}_i}((\nabla')^{\cdot})$;
(ii) $\nabla' \vdash_{\mathbb{T}} \overline{a}_i \# t_i : \mathsf{s}_i$ (see (9));
(iii) $[t_i]$ is the equivalence class of terms u such that $\nabla' \vdash_{\mathbb{T}} t_i \approx u : \mathsf{s}_i$.

Identity: *The identity on* ∇ (15) *is*

$$id_\nabla \;\triangleq\; \nabla \vdash (\overline{a}_1 \mathrel{\#} [v_1] : \mathsf{s}_1, \ldots, \overline{a}_n \mathrel{\#} [v_n] : \mathsf{s}_n) \;\; .$$

Composition: *Given* f (16), $g = \nabla \vdash (\overline{a}'_1 \mathrel{\#} [t'_1] : \mathsf{s}'_1, \ldots, \overline{a}'_m \mathrel{\#} [t'_m] : \mathsf{s}'_m)$,

$$g \circ f \;\triangleq\; \nabla' \vdash (\overline{a}'_1 \mathrel{\#} [t'_1\{\sigma\}] : \mathsf{s}'_1, \ldots, \overline{a}'_m \mathrel{\#} [t'_m\{\sigma\}] : \mathsf{s}'_m)$$

where σ *is the substitution* (5) $\sigma(v_i) = t_i$ *for* $1 \leq i \leq n$.

Finitely supported internal Perm**-action:** *Given* ∇ (15),

$$\pi_\nabla \;\triangleq\; \nabla \vdash (\pi \cdot \overline{a}_1 \mathrel{\#} [\pi\, v_1] : \pi \cdot \mathsf{s}_1, \ldots, \pi \cdot \overline{a}_n \mathrel{\#} [\pi\, v_n] : \pi \cdot \mathsf{s}_n) \;\; .$$

Equivariant finite products: *The terminal object of* $Cl(\mathbb{T})$ *is the empty freshness environment. Given* ∇ (15) *and* $\nabla' = (\overline{a}'_1 \mathrel{\#} v_1 : \mathsf{s}'_1, \ldots, \overline{a}'_m \mathrel{\#} v_m : \mathsf{s}_m)$, *their binary product* $\nabla \times \nabla'$ *is*

$$(\overline{a}_1 \mathrel{\#} v_1 : \mathsf{s}_1, \ldots, \overline{a}_n \mathrel{\#} v_n : \mathsf{s}_n,\, \overline{a}'_1 \mathrel{\#} v_{n+1} : \mathsf{s}'_1, \ldots, \overline{a}'_m \mathrel{\#} v_{n+m} : \mathsf{s}'_m)$$

with projections

$$\begin{aligned} pr_1 &\triangleq \nabla \times \nabla' \vdash (\overline{a}_1 \mathrel{\#} [v_1] : \mathsf{s}_1, \ldots, \overline{a}_n \mathrel{\#} [v_n] : \mathsf{s}_n) \;\; ; \\ pr_2 &\triangleq \nabla \times \nabla' \vdash (\overline{a}'_1 \mathrel{\#} [v_{n+1}] : \mathsf{s}'_1, \ldots, \overline{a}'_m \mathrel{\#} [v_{n+m}] : \mathsf{s}'_m) \;\; . \end{aligned}$$

Fresh subobjects: *Define* $\nabla^{\#\overline{a}}$ *by applying* (7) *to* (15). *Then*

$$i^{\overline{a}}_\nabla \;\triangleq\; \nabla^{\#\overline{a}} \vdash (\overline{a}_1 \mathrel{\#} [v_1] : \mathsf{s}_1, \ldots, \overline{a}_n \mathrel{\#} [v_n] : \mathsf{s}_n) \;\; .$$

Proof. App. A.2.

Definition 6.2. *Define the* generic algebra G *by*

$$\begin{aligned} G[\![\mathsf{s}]\!] &\triangleq (v_1 : \mathsf{s}) \\ G[\![op]\!] &\triangleq (v_1 : \mathsf{s}_1, \ldots, v_n : \mathsf{s}_n) \vdash (\emptyset \mathrel{\#} [op\, v_1 \cdots v_n] : \mathsf{s}) \end{aligned}$$

for $op : (\mathsf{s}_1, \ldots, \mathsf{s}_n) \to \mathsf{s}$. *It is easy to prove that* G *is a* $\mathbb{T}$*-algebra in* $Cl(\mathbb{T})$.

Theorem 6.3 (Completeness). *Given a NEL-theory* $\mathbb{T}$, *if* $\nabla \vdash t \approx t' : \mathsf{s}$ *is satisfied by all* $\mathbb{T}$*-algebras in all nominal Lawvere theories then* $\nabla \vdash_{\mathbb{T}} t \approx t' : \mathsf{s}$.

Proof. If $\nabla \vdash t \approx t' : \mathsf{s}$ is satisfied by the generic algebra G in $Cl(\mathbb{T})$ then $\nabla \vdash_{\mathbb{T}} t \approx t' : \mathsf{s}$ by the definition of $Cl(\mathbb{T})$-arrows.

Theorem 6.4. *Given any NEL-theory* $\mathbb{T}$ *and nominal Lawvere theory* $\mathcal{C}$, *there is an isomorphism*

$$FM(Cl(\mathbb{T}), \mathcal{C}) \cong \mathcal{C}^{\mathbb{T}}$$

between the category of FM-functors $Cl(\mathbb{T}) \to \mathcal{C}$ *(Def. 4.10) and the category of* $\mathbb{T}$*-algebras in* $\mathcal{C}$ *(Def. 5.5).*

Proof. App. A.2.

Definition 6.5. *Given a nominal Lawvere theory $\mathcal{C}$, define the signature $Sg(\mathcal{C})$ by setting $\mathsf{Sort}_{Sg(\mathcal{C})} = ob\,\mathcal{C}$ and*

$$\mathsf{Op}_{Sg(\mathcal{C})} \triangleq \{f : (C_1, \ldots, C_n) \to C \mid f : C_1 \times \cdots \times C_n \to C \in ar\,\mathcal{C}\}$$

with Perm*-actions defined via the internal* Perm*-action on $\mathcal{C}$. We use smallness here as our sorts and operation symbols must form nominal* sets. *Note that one arrow can give rise to multiple operation symbols; for example, $f : C_1 \times C_2 \to C$ induces operation symbols $(C_1 \times C_2) \to C$ and $(C_1, C_2) \to C$.*

Let $M(\mathcal{C})$ be the $Sg(\mathcal{C})$-structure in $\mathcal{C}$ which we define by $M(\mathcal{C})[\![C]\!] = C$ and $M(\mathcal{C})[\![f]\!] = f$, then let $Th(\mathcal{C})$ be the $Sg(\mathcal{C})$-theory whose axioms are all the judgements that are satisfied by $M(\mathcal{C})$, so $M(\mathcal{C})$ is trivially an algebra of $\mathcal{C}^{Th(\mathcal{C})}$.

Remark 6.6. It is clear that nominal Lawvere theories require a Perm-action on their objects; without this we could not adequately represent freshness environments in the classifying category. The translation from nominal Lawvere theories back to NEL-theories, if it is not to lose information, then requires that our sorts may form any nominal set. Note that this is only a mild generalisation of previous presentations of NEL [5], as by Ex. 2.2(i) any set may be considered a nominal set under the trivial Perm-action.

Theorem 6.7. *For any small FM-category $\mathcal{C}$ there is an equivalence*

$$\mathcal{C} \;\simeq\; Cl(Th(\mathcal{C}))\ .$$

Proof. App. A.2.

Proving the converse of this theorem, that $\mathbb{T} \simeq Th(Cl(\mathbb{T}))$, requires a notion of morphism between theories. The most natural definition of such a translation $\mathbb{T} \to \mathbb{T}'$ is an FM-functor $Cl(\mathbb{T}) \to Cl(\mathbb{T}')$, so the converse of Thm. 6.7 is actually a corollary, and we have a correspondence between NEL-theories and nominal Lawvere theories.

7 Related and Further Work

- This work opens the way for consideration of NEL-theories in FM-categories other than $\mathcal{FM}$-Set, such as the FM-cppos of [21].
- The most similar system to NEL is the independently produced Nominal Algebra (NA) [10], which also addresses languages with binding, and equations modulo freshness, via the nominal sets model. A number of different design choices were made in these logics' constructions: NA employs sets, rather than nominal sets, of sorts and operation symbols; NA uses 'nominal signatures' with explicit binding sorts; and freshness in NA is sound, but not complete, for freshness in the underlying nominal sets interpretation. Nonetheless it is not too difficult to translate from one logic to the other, as is done to some extent in [10, Sec. 5] and [4, Sec. 7].

The motivation for many of the design choices made for NEL was to cleave as close as possible to the standard account of equational logic, so that established equational logic techniques may be transferred to the nominal setting. This paper is one of the fruits of this philosophy. While nominal Lawvere theories obviously apply to NA-theories if we translate them to NEL-theories as our first step, a more interesting open question is whether a compelling Lawvere theoretic account can be developed directly for some of the design choices of NA, most notably explicit binding sorts.

- The results of this paper are unlikely to be a straightforward application of existing work on generalised Lawvere theories [19,14], concerned as it is with expressing the particular syntax of NEL. Nonetheless, connecting nominal Lawvere theories with the more general picture would be valuable.
- The most well known alternative to Lawvere theories for the category theoretic expression of equational logic are algebras for a monad. This was investigated for NEL in [3, Chap. 6], and more extensively in [13]. Unlike Lawvere theories, the monadic view offers no explicit category theoretic description of a logic's theories.
- Fiore-Hur equational systems are a newer category theoretic approach to equational logic, and have an established application to NEL [8, Sec. 7.3]. There is as yet no general relationship established between this approach and Lawvere theories, but such results have been produced in the special case of second-order equational logic [7].

References

1. Aguiar, M.: Internal categories and quantum groups. Ph.D. thesis, Cornell University (1997)
2. Cartmell, J.: Generalised algebraic theories and contextual categories. Ann. Pure Appl. Logic 32, 209–243 (1986)
3. Clouston, R.: Equational Logic for Names and Binders. Ph.D. thesis, University of Cambridge (2009), http://cecs.anu.edu.au/~rclouston/Clouston_Thesis.pdf
4. Clouston, R.: Binding in nominal equational logic. ENTCS 265, 259–276 (2010)
5. Clouston, R., Pitts, A.M.: Nominal equational logic. ENTCS 172, 223–257 (2007)
6. Crole, R.L.: Categories for Types. Cambridge University Press, Cambridge (1993)
7. Fiore, M., Mahmoud, O.: Second-order algebraic theories. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 368–380. Springer, Heidelberg (2010)
8. Fiore, M., Hur, C.K.: On the construction of free algebras for equational systems. TCS 410(18), 1704–1729 (2009)
9. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. FAC 13, 341–363 (2002)
10. Gabbay, M.J., Mathijssen, A.: Nominal (universal) algebra: equational logic with names and binding. J. Logic Comput. 19(6), 1455–1508 (2009)
11. Hyland, M., Power, J.: The category theoretic understanding of universal algebra: Lawvere theories and monads. ENTCS 172, 437–458 (2007)
12. Johnstone, P.L.: Sketches of an Elephant: A Topos Theory Compendium, vol. 1. Oxford University Press, Oxford (2002)
13. Kurz, A., Petrişan, D.: On universal algebra over nominal sets. MSCS 20(2), 285–318 (2010)

14. Lack, S., Rosický, J.: Notions of Lawvere theory. Appl. Categ. Structures (2009) (to appear)
15. Lawvere, F.W.: Functorial Semantics of Algebraic Theories. Ph.D. thesis, Columbia University (1963)
16. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes. I. Inform. and Comput. 100, 1–40 (1992)
17. Pitts, A.M.: Nominal logic: A first order theory of names and binding. In: Kobayashi, N., Babu, C. S. (eds.) TACS 2001. LNCS, vol. 2215, pp. 219–242. Springer, Heidelberg (2001)
18. Plotkin, G.D.: Some varieties of equational logic. In: Futatsugi, K., Jouannaud, J.-P., Bevilacqua, V. (eds.) Algebra, Meaning, and Computation. LNCS, vol. 4060, pp. 150–156. Springer, Heidelberg (2006)
19. Power, J.: Enriched Lawvere theories. TAC 6(7), 83–93 (1999)
20. Schröder, L.: Classifying categories for partial equational logic. ENTCS 69, 305–322 (2003)
21. Shinwell, M.R., Pitts, A.M.: On a monadic semantics for freshness. TCS 342, 28–55 (2005)

A Technical Appendices

A.1 Algebra in FM-Categories

Lemma A.1. *(i)* $\llbracket \nabla \vdash \pi * t : \pi \cdot \mathsf{s} \rrbracket = \pi_{\llbracket \mathsf{s} \rrbracket} \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket$*;*

(ii) Given $\nabla \leq \nabla'$ *there exists an arrow* $\mathsf{weak} : \llbracket \nabla' \rrbracket \to \llbracket \nabla \rrbracket$ *such that for any* $t \in \Sigma_{\mathsf{s}}(\nabla^{:})$, $\llbracket \nabla' \vdash t : \mathsf{s} \rrbracket = \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ \mathsf{weak}$*;*

(iii) $\llbracket \nabla^{\#\overline{a}} \vdash t : \mathsf{s} \rrbracket = \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{\overline{a}}_{\llbracket \nabla \rrbracket}$*;*

(iv) If ∇ *is as* (6) *then* $\llbracket \nabla \vdash t : \mathsf{s} \rrbracket = \llbracket x_1 : \mathsf{s}_1, \ldots, x_n : \mathsf{s}_n \vdash t : \mathsf{s} \rrbracket \circ (i^{\overline{a}_1}_{\llbracket \mathsf{s}_1 \rrbracket} \times \cdots \times i^{\overline{a}_n}_{\llbracket \mathsf{s}_n \rrbracket})$*;*

(v) Given $\Gamma \in \mathsf{SE}_{\Sigma}$ *with domain* $\{x_1, \ldots, x_n\}$*, a substitution* $\sigma : \Gamma \to \nabla^{:}$ *and a term* $t \in \Sigma_{\mathsf{s}}(\Gamma)$*,*

$$\llbracket \nabla \vdash t\{\sigma\} : \mathsf{s} \rrbracket = \llbracket \Gamma \vdash t : \mathsf{s} \rrbracket \circ \langle \ldots, \llbracket \nabla \vdash \sigma(x_i) : \Gamma(x_i) \rrbracket, \ldots \rangle$$

where Γ *is defined to be a freshness environment in* FE_{Σ} *by* $\Gamma(x) = (\Gamma(x), \emptyset)$*.*

Proof. (i) follows by induction on the structure of t, using the naturality of π in the constructed term chase.

(ii): take ∇ as (6) and say $\nabla'(x_i) = (\overline{a}_i \cup \overline{a}'_i, \mathsf{s}_i)$ for $1 \leq i \leq n$. Then set

$$\mathsf{weak} \;\triangleq\; \langle i^{\overline{a}'_1} \circ pr_1, \ldots, i^{\overline{a}'_n} \circ pr_n \rangle \enspace .$$

The result follows by another induction on t, and (iii) is an immediate corollary. (iv) is another induction on t and (v) likewise, using (i) in the suspension case.

Definition A.2. *Suppose* $\nabla \vdash \overline{a} \# t : \mathsf{s}$ *is satisfied, so by* (9) *and Def. 5.3,* $\llbracket \nabla^{\# supp(\vec{a}')} \vdash t : \mathsf{s} \rrbracket = \llbracket \nabla^{\# supp(\vec{a}')} \vdash (\vec{a}\;\vec{a}') * t : \mathsf{s} \rrbracket$*. Then by Lem. A.1(i) and (iii),*

$\llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{supp(\vec{a}')} = (\vec{a}\ \vec{a}') \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{supp(\vec{a}')}$. *By Def. 4.5(vii) a unique arrow is induced which we will call* $\llbracket \nabla \vdash \overline{a} \# t : \mathsf{s} \rrbracket$*:*

$$\begin{tikzcd}
\llbracket \nabla \rrbracket^{\# supp(\vec{a}')} \arrow[r, "i^{supp(\vec{a}')}"] & \llbracket \nabla \rrbracket \arrow[d, dotted, "\llbracket \nabla \vdash \overline{a} \# t : \mathsf{s} \rrbracket"'] \arrow[dr, "\llbracket \nabla \vdash t : \mathsf{s} \rrbracket"] & \\
& \llbracket \mathsf{s} \rrbracket^{\# \overline{a}} \arrow[r, "i^{\overline{a}}"'] & \llbracket \mathsf{s} \rrbracket \arrow[loop right, "(\vec{a}\ \vec{a}')"]
\end{tikzcd}$$

Lemma A.3. *Take* ∇ *as* (6).

(i) Given a term $t \in \Sigma_{\mathsf{s}}(\nabla^{\cdot})$ *and substitution* $\sigma : \nabla^{\cdot} \to (\nabla')^{\cdot}$, *if the arrows* $\llbracket \nabla' \vdash \overline{a}_i \# \sigma(x_i) : \mathsf{s}_i \rrbracket$ *are defined for* $1 \leq i \leq n$ *then* $\llbracket \nabla' \vdash t\{\sigma\} : \mathsf{s} \rrbracket$ *equals*

$$\llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ \langle \llbracket \nabla' \vdash \overline{a}_1 \# \sigma(x_1) : \mathsf{s}_1 \rrbracket, \ldots, \llbracket \nabla' \vdash \overline{a}_n \# \sigma(x_n) : \mathsf{s}_n \rrbracket \rangle \ ;$$

(ii) $id_{\llbracket \nabla \rrbracket} = \langle \llbracket \nabla \vdash \overline{a}_1 \# x_1 : \mathsf{s}_1 \rrbracket, \ldots, \llbracket \nabla \vdash \overline{a}_n \# x_n : \mathsf{s}_n \rrbracket \rangle$;

(iii) $i^{\overline{a}}_{\llbracket \nabla \rrbracket} = \langle \llbracket \nabla^{\# \overline{a}} \vdash \overline{a}_1 \# x_1 : \mathsf{s}_1 \rrbracket, \ldots, \llbracket \nabla^{\# \overline{a}} \vdash \overline{a}_n \# x_n : \mathsf{s}_n \rrbracket \rangle$;

(iv) $\pi_{\llbracket \nabla \rrbracket} = \langle \llbracket \nabla \vdash \pi \cdot \overline{a}_1 \# \pi x_1 : \pi \cdot \mathsf{s}_1 \rrbracket, \ldots, \llbracket \nabla \vdash \pi \cdot \overline{a}_n \# \pi x_n : \pi \cdot \mathsf{s}_n \rrbracket \rangle$;

(v) Take ∇_1, ∇_2 *with disjoint domain. Then* $pr_j(\llbracket \nabla_1 \rrbracket, \llbracket \nabla_2 \rrbracket) = \langle \llbracket \nabla_1 \cup \nabla_2 \vdash \overline{a}_1 \# x_1 : \mathsf{s}_1 \rrbracket, \ldots, \llbracket \nabla_1 \cup \nabla_2 \vdash \overline{a}_n \# x_n : \mathsf{s}_n \rrbracket \rangle$ *if* ∇_j *is* (6) *for* $i = 1$ *or* 2.

Proof. (i) follows by Lem. A.1(iv) and (v). For (ii)-(v) we first need to confirm that the arrows in question exist, following Def A.2. For (ii), we see that

$$i^{\overline{a}_i} \circ pr_i \circ i^{supp(\vec{a}'_i)} \ = \ (\vec{a}_i\ \vec{a}'_i) \circ i^{\overline{a}_i} \circ pr_i \circ i^{supp(\vec{a}'_i)}$$

by the naturality of $(\vec{a}_i\ \vec{a}'_i)$ and Def. 4.5(v). (iii)-(v) follow similarly. The equalities (ii), (iii) and (v) then follow by applying $i^{\overline{a}_i} \circ pr_i$ to each side, as $i^{\overline{a}_i}$ is mono and projections jointly mono. (iv) follows by applying $i^{\pi \cdot \overline{a}_i} \circ pr_i$ to each side.

*Proof (**Thm. 5.4**).* Closure under (REFL), (SYMM) and (TRANS) is trivial, and (WEAK) follows from Lem A.1(ii).

(SUBST): Given ∇ as (6) the arrows $\llbracket \nabla' \vdash \overline{a}_1 \# \sigma(x_1) : \mathsf{s}_1 \rrbracket$ (Def. A.2) are defined for $1 \leq i \leq n$, so by Lem. A.3(i)

$$\llbracket \nabla' \vdash t\{\sigma\} : \mathsf{s} \rrbracket \ = \ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ \langle \llbracket \nabla' \vdash \overline{a}_1 \# \sigma(x_1) : \mathsf{s}_1 \rrbracket, \ldots \rangle$$

and similarly for $t'\{\sigma'\}$. We have $\llbracket \nabla \vdash t : \mathsf{s} \rrbracket = \llbracket \nabla \vdash t' : \mathsf{s} \rrbracket$, while $i^{\overline{a}_i} \circ \llbracket \nabla' \vdash \overline{a}_i \# \sigma(x_i) : \mathsf{s}_i \rrbracket = \llbracket \nabla' \vdash \sigma(x_i) : \mathsf{s}_i \rrbracket = \llbracket \nabla' \vdash \sigma'(x_i) : \mathsf{s}_i \rrbracket = i^{\overline{a}_i} \circ \llbracket \nabla' \vdash \overline{a}_i \# \sigma'(x_i) : \mathsf{s}_i \rrbracket$. But $i^{\overline{a}_i}$ is mono by Lem. 4.9, so we are done.

(ATM-ELIM): $\llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{\overline{a}} = \llbracket \nabla \vdash t' : \mathsf{s} \rrbracket \circ i^{\overline{a}}$ by Lem. A.1(iii), so $\llbracket \nabla \vdash t : \mathsf{s} \rrbracket = \llbracket \nabla \vdash t' : \mathsf{s} \rrbracket$ by Def. 4.5(vi).

(PERM): By Lemma A.1(i) and (iii) we need to prove that $\pi \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')} = \pi' \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')}$. Now $\pi^{-1} \circ \pi' \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')} = \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')} \circ \pi^{-1}\pi'_{\llbracket \nabla \rrbracket^{\# ds(\pi,\pi')}}$ by naturality, but $ds(\pi, \pi') = supp(\pi^{-1}\pi')$, so we can apply Def. 4.5(v) to make this equal $\llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')}$. Applying the identity to the front gives us $\pi^{-1} \circ \pi \circ \llbracket \nabla \vdash t : \mathsf{s} \rrbracket \circ i^{ds(\pi,\pi')}$. π^{-1} is iso, and therefore mono, so we are done.

A.2 Category-Theory Correspondence

*Proof (**Lem. 6.1**).* The various properties of FM-categories are easily verifiable corollaries of the proof rules, along with standard properties of NEL-terms from [5, Sec. 4 and 5]. For example, take f as (16) and apply (10):

$$\pi \cdot f \ = \ \pi \cdot \nabla' \vdash [\pi \cdot \overline{a}_1 \not\# [\pi \cdot t_1] : \pi \cdot \mathsf{s}_1, \ldots, \pi \cdot \overline{a}_n \not\# [\pi \cdot t_n] : \pi \cdot \mathsf{s}_n] \ .$$

$\pi \cdot t$ is the *meta-level* Perm-*action* of [5, Sec. 5] (not to be confused with the *object-level* Perm-*action* (4)) under which terms are indeed finitely supported.

Likewise, if we take f as (16) in (12) then

$$\hat{f} \ = \ \nabla' \vdash [\overline{a}_1 \cup \overline{a} \not\# [t_1] : \mathsf{s}_1, \ldots, \overline{a}_n \cup \overline{a} \not\# [t_n] : \mathsf{s}_n] \ .$$

Lemma A.4. *Given FM-categories $\mathcal{C}, \mathcal{C}'$ and an algebra $M \in ob\,\mathcal{C}^{\mathbb{T}}$, we can define a functor, called the* modelling functor*, $M(-) : FM(\mathcal{C}, \mathcal{C}') \to \mathcal{C}^{\mathbb{T}}$ by*

- *$M(F)[\![\mathsf{s}]\!] = F(M[\![\mathsf{s}]\!])$ and $M(F)[\![op]\!] = F(M[\![op]\!])$;*
- *$M(\phi)_{\mathsf{s}} = \phi_{M[\![\mathsf{s}]\!]}$.*

Proof. FM-functors preserve the internal Perm-action, so $\mathsf{s} \mapsto M(F)[\![\mathsf{s}]\!]$ and $op \mapsto M(F)[\![op]\!]$ are equivariant and $M(F)$ is a Σ-structure. Given $t \in \Sigma_{\mathsf{s}}(\nabla^{:})$,

$$M(F)[\![\nabla \vdash t : \mathsf{s}]\!] \ = \ F(M[\![\nabla \vdash t : \mathsf{s}]\!]) \tag{17}$$

by induction on the structure of t, so if $M \in ob\,\mathcal{C}^{\mathbb{T}}$ then $M(F) \in ob\,\mathcal{C}'^{\mathbb{T}}$. $\mathsf{s} \mapsto M(\phi)_{\mathsf{s}}$ is evidently equivariant, and (14) holds because natural transformations between finite product preserving functors commute with those products.

Lemma A.5. *Given ∇ as (15), a $\mathbb{T}$-homomorphism $h : M \to M'$, and a term $t \in \Sigma_{\mathsf{s}}(\nabla^{:})$,*

$$h_{\mathsf{s}} \circ M[\![\nabla \vdash t : \mathsf{s}]\!] = M'[\![\nabla \vdash t : \mathsf{s}]\!] \circ (h_{\mathsf{s}_1}^{\not\#\overline{a}_1} \times \cdots \times h_{\mathsf{s}_n}^{\not\#\overline{a}_n})$$

Proof. An easy induction on the structure of t.

*Proof (**Thm. 6.4**).* We will show that the modelling functor (Lem. A.4) for the generic algebra (Def. 6.2) is an isomorphism $G(-) : FM(Cl(\mathbb{T}), \mathcal{C}) \to \mathcal{C}^{\mathbb{T}}$. Let $G^{-1}(-) : \mathcal{C}^{\mathbb{T}} \to FM(Cl(\mathbb{T}), \mathcal{C})$ be

$$\begin{aligned} G^{-1}(M)(\nabla) &= M[\![\nabla]\!] \\ G^{-1}(M)(f) &= \langle M[\![\nabla' \vdash \overline{a}_1 \not\# t_1 : \mathsf{s}_1]\!], \ldots, M[\![\nabla' \vdash \overline{a}_n \not\# t_n : \mathsf{s}_n]\!]\rangle \\ G^{-1}(h)_{\nabla} &= h_{\mathsf{s}_1}^{\not\#\overline{a}_1} \times \cdots \times h_{\mathsf{s}_n}^{\not\#\overline{a}_n} \end{aligned} \tag{18}$$

where ∇ is (15), $f : \nabla' \to \nabla$ is (16) and $h_{\mathsf{s}_i}^{\not\#\overline{a}_i}$ is defined by Def. 4.8. For any $\mathbb{T}$-algebra M in $\mathcal{C}$, $G^{-1}(M)$ is an FM-functor $Cl(\mathbb{T}) \to \mathcal{C}$ by Lem. A.3.

Given a $\mathbb{T}$-homomorphism $h : M \to M'$, we can show that $i^{\overline{a}_i} \circ pr_i \circ G^{-1}(M')(f) \circ G^{-1}(h)_{\nabla'} = i^{\overline{a}_i} \circ pr_i \circ G^{-1}(h)_{\nabla} \circ G^{-1}(M)(f)$ by Lem. A.5 and Defs. 4.8 and A.2. But $i^{\overline{a}_i} \circ pr_i$ are jointly mono, so $G^{-1}(h)$ is a natural transformation $G^{-1}(M) \dot{\to} G^{-1}(M')$.

That $G(G^{-1}(-))$ is the identity on $\mathcal{C}^{\mathbb{T}}$ follows easily; the converse holds as follows. Given an FM-functor $F : Cl(\mathbb{T}) \to \mathcal{C}$ and $Cl(\mathbb{T})$-object ∇ we have $G^{-1}(G(F))(\nabla) = G(F)[\![\nabla]\!] = F(G[\![\nabla]\!]) = F\nabla$ because F preserves finite products and $G[\![\nabla]\!] = \nabla$. $i^{\overline{a}_i} \circ pr_i \circ G^{-1}(G(F))(f) = F(G[\![\nabla' \vdash t_i : \mathsf{s}_i]\!])$ by (17), which equals $F(\nabla \vdash ([t_i] : \mathsf{s}_i)) = i^{\overline{a}_i} \circ pr_i \circ f$. But $i^{\overline{a}_i} \circ pr_i$ are jointly mono so we have equality on objects. Finally, given a natural transformation $\phi : F \dot{\to} F'$ we must show that $G^{-1}(G(\phi))_\nabla = \phi_\nabla$. This follows by applying $i^{\overline{a}_i} \circ pr_i$ as above.

*Proof (**Thm. 6.7**).* The equivalence functor is $G^{-1}(M(\mathcal{C})) : Cl(Th(\mathcal{C})) \to \mathcal{C}$, defined by Def. 6.5 and (18).

Full: Take the $\mathcal{C}$-arrow $f : G^{-1}(M(\mathcal{C}))(\nabla') \to G^{-1}(M(\mathcal{C}))(\nabla)$. Then $(v_1 : G^{-1}(M(\mathcal{C}))(\nabla')) \vdash ([f\, v_1] : G^{-1}(M(\mathcal{C}))(\nabla))$ is a $Cl(Th(\mathcal{C}))$-arrow. Applying $G^{-1}(M(\mathcal{C}))$ to this arrow gives us $M(\mathcal{C})[\![f]\!] = f$.

Faithful: Say we have $Cl(Th(\mathcal{C}))$-arrows $f = \nabla' \vdash (\overline{a}_1 \not\#\, [t_1] : C_1, \ldots)$ and $f' = \nabla' \vdash (\overline{a}_1 \not\#\, [t'_1] : C_1, \ldots)$ so that $G^{-1}(M(\mathcal{C}))(f) = G^{-1}(M(\mathcal{C}))(f')$. Applying the jointly mono $i^{\overline{a}_i} \circ pr_i$ to each side gives us $M(\mathcal{C})[\![\nabla' \vdash t_i : C_i]\!] = M(\mathcal{C})[\![\nabla' \vdash t'_i : C_i]\!]$, so $\nabla' \vdash_{Th(\mathcal{C})} t_i \approx t'_i : C_i$ by definition and $f = f'$.

Surjective: For any $\mathcal{C}$-object C, $G^{-1}(M(\mathcal{C}))(v_1 : C) = C$.

Turing Machines on Cayley Graphs

Aubrey da Cunha

University of Michigan, Ann Arbor, MI 48109, USA

Abstract. We present a generalization of standard Turing machines based on allowing unusual tapes. We present a set of reasonable constraints on tape geometry and conclude that the proper degree of generality is Cayley graphs. Surprisingly, this generalization does not lead to yet another equivalent formulation of the notion of computable function. Rather, it gives an alternative definition of the recursively enumerable Turing degrees that does not rely on oracles.

1 Introduction

When Alan Turing originally defined his *a*-machines, which would later be called Turing machines, he envisioned a machine whose memory was laid out along a one-dimensional tape, inspired by the ticker tapes of the day. This seemed somewhat arbitrary and perhaps unduly restrictive, and so, very quickly, machines with multiple and multi-dimensional tapes were proposed. The focus at the time was defining the term "computable", and as adding tapes and dimensions defined the same class of functions as Turing's original, simpler model, studying alternate tape geometries fell out of favor for some time.

The complexity theory community then reignited interest in alternative tape geometries by considering not the class of functions computable by Turing machines, but time and space complexity of functions on different tape geometries. This led to a number of results about relative efficiency of machines with one/many tapes and one/two/high-dimensional tapes. For example, the language of palindromes can be computed in $O(n)$ time on a two-tape machine, but requires $\Omega(n^2)$ time on a one-tape machine with one read/write head. Or, an m-dimensional Turing machine running in time $T(n)$ can be simulated by a k-dimensional Turing machine ($k < m$) in time $T(n)^{1+\frac{1}{m}-\frac{1}{k}+\epsilon}$ for all $\epsilon > 0$ [4].

Many of the proofs and algorithms used in the study of multidimensional Turing machines make their way into or are inspired by the world of mesh-connected systems. Mesh-connected systems are arrays of identical, relatively dumb processors (typically with memory logarithmic in the input size) that communicate with their neighbors to perform a computation. Time use on a Turing machine with a d-dimensional tape is intimately tied to the power consumption of a d-dimensional mesh of processors with finite memory, so there is some natural crossover. Mesh-connected systems constitute an area of very active research now, but since it remains very closely tied to the physical implementation, research is generally restricted to two- and three-dimensional grids.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 84–94, 2011.

In this paper, we go beyond the world of rectangular grids and consider tapes at their most general. The purpose of this is two-fold. First, to give the complexity theorists a general framework in which to work, subsuming all current tape-based Turing models. Second, to provide some evidence that alternative tape geometries are interesting from a recursion-theoretic perspective. Along the way, we will encounter some questions about well-orderings in finitely branching trees that are interesting in their own right.

Any Turing tape can be modeled as a digraph with nodes corresponding to tape cells and edges corresponding to allowable transitions. Hence, we start by introducing a number of graph-theoretic conditions that ought to be satisfied by a reasonable tape. It turns out that the criteria we present are necessary and sufficient conditions for the graph to be the Cayley graph of a finitely generated, infinite group.

We then turn to the question of whether allowing arbitrary Cayley graphs as tapes is just another equivalent machine model for the class of computable functions. Interestingly, this will depend on the structure of the group from which the Cayley graph is generated. For groups with solvable word problem, this does indeed lead to machines that compute the class of computable functions, however for groups with unsolvable word problems, these machines are strictly more powerful than standard Turing machines. In fact, they can be as powerful as any oracle machine and we end up with an alternative definition of the Turing degrees that is machine based and doesn't rely on oracles.

Throughout the rest of this paper, all Turing machines will have a single tape and a single head. This is the easiest case to treat and the generalization to multiple tapes and multiple heads is entirely analogous to the standard Turing picture. We will also restrict ourselves to a deterministic setting for the same reasons.

2 General Tape Geometries

Any tape essentially consists of a collection of cells, each of which can hold a single symbol, together with a mechanism for moving from one cell to another. So underlying any tape is an edge-colored digraph. Edges in this graph represent allowable transitions between states and the edge coloring encodes the conditions on the stored symbol and control state under which that particular transition occurs. In order to be a reasonable Turing tape, this digraph should satisfy a few restrictions.

1. Uniqueness of outgoing colors - From any vertex, there should be exactly one outgoing edge of each color. Since the mechanism by which the tape head moves is encoded in the edge color, outgoing edges should have different colors. Also, since the transition function is independent of the tape cell, the collection of colors going out from each vertex should be the same.
2. Homogeneity - Every vertex should "look like" every other vertex. Technically, this means that the subgroup of the automorphism group of the graph that preserves edge colors should be vertex transitive. This is an extension

of the assumption that all tape cells are indistinguishable, in this case by the local geometry.

3. Infinity - The tape should have an infinite number of cells. Otherwise, it's just a finite automaton.
4. Connectedness - As the head moves during the course of a computation, it remains on a single connected component. Inaccessible cells are useless, so we can require that every tape cell be accessible from the starting point. In particular, this means that the graph is connected.
5. Finitely many colors - The transition function of the TM should be finite, so there should only be finitely many outgoing colors. Having more colors doesn't change the computational power, since only finitely many of them could be referenced by the transition function anyway.
6. Backtracking - The Turing machine should be able to return to the cell it just came from. This assumption is less essential, since in view of homogeneity, any algorithm that called for returning to the previous tape cell could be replaced by a fixed sequence of steps. However, many algorithms call for the head to return to the previous cell and forcing the head to do so by a circuitous route seems unduly harsh. Note that in view of homogeneity, the color of an edge determines the color of the reverse edge.

Restrictions 1 and 2 imply that our tape is a Cayley graph, restrictions 3, 4, and 5 make it the Cayley graph of a finitely generated infinite group, and restriction 6 forces the generating set to be closed under inverses. In addition, any Cayley graph of a finitely generated infinite group with a generating set that is closed under inverses satisfies 1–6. This suggests that Cayley graphs are, in some sense, the "right" degree of generality and leads to the following definition:

Definition 1. *Let G be an infinite group and $S \subset G$ be a finite generating set for G that is closed under inverses. Then the Cayley graph of G generated by S is called the* tape graph*, (G, S).*

Using this general type of tape, we can then ask questions about the structure of Turing Machines with tapes given by assorted groups and generating sets.

3 Turing Machines on Cayley Graphs

Definition 2. *Let $G = \langle g_1, \ldots, g_n \rangle$ be a finitely generated group with the set $\{g_1, \ldots, g_n\}$ closed under inverses. Then a Turing Machine over G with generating set $\langle g_1, \ldots, g_n \rangle$ is a 7-tuple, $(Q, \Gamma, b, \Sigma, \delta, q_0, F)$ where*

- *Q is the finite set of states*
- *Γ is the finite set of tape symbols*
- *$b \in \Gamma$ is a designated blank symbol*
- *$\Sigma \subset \Gamma \backslash \{b\}$ is the set of input symbols*
- *$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{g_1, \ldots, g_n\}$ is the transition function*
- *$q_0 \in Q$ is the initial state*
- *$F \subset Q$ is the set of terminal states (typically one to accept and one to reject)*

This definition varies from the standard definition only in the interpretation of the transition function. Whereas a standard TM has a two-way infinite one-dimensional tape and the transition function includes instructions for moving left or right, a TM over G has as a tape the Cayley graph of G and the transition function has instructions for moving along edges labeled by a particular generator. For example, a Turing Machine over $\mathbb{Z}$ with generating set $\{-1, +1\}$ is a standard one-dimensional TM and a TM over $\mathbb{Z}^2$ with generating set $\{(0, -1), (-1, 0), (0, 1), (1, 0)\}$ is a standard two-dimensional TM.

We cannot yet describe the behavior of such a machine as a computation, since computation is generally done on strings and we have as yet no way to encode strings into machine states. For standard one-dimensional machines, this is done in a straightforward manner, but for more exotic tape geometries, there isn't one obviously correct way to provide the machine with input. This can be done in a canonical way, but it requires some machinery.

3.1 A Well-Ordering in Trees

We shall turn aside from the main topic for a moment to discuss a general statement about trees. There are many ways to define an order on the vertices of a tree, but we are going to be interested in the lexicographic order. In general, lexicographic orderings on trees have few nice properties, but we show that finitely branching trees have a subtree where the lexicographic order is in fact a well-order.

First, some definitions. Let T be a finitely branching tree. Denote by $[T]$ the set of all infinite paths through T (our paths begin at the root and follow the child relation) and by $\sqsubseteq$ the partial ordering on vertices induced by the tree. Our convention will be that $v \sqsubseteq w$ means that v is closer to the root than w.

In order for the lexicographic ordering on T to even make sense, we must have a linear order on the set of children of each node. Denote the order on the children of $v \in T$ by $\leq_v$. Then the lexicographic order, $\leq$ is defined as follows:

- If $v \sqsubseteq w$, then $v \leq w$.
- If neither $v \sqsubseteq w$ nor $w \sqsubseteq v$, let u be the greatest lower bound of v and w according to $\sqsubseteq$ and let $u' \sqsubseteq v$ and $u'' \sqsubseteq w$ be children of u. Then $v \leq w$ if and only if $u' \leq_u u''$.

This order can, in fact, be extended to an order on $T \cup [T]$. Identifying elements of $[T]$ with subsets of T and elements of T with one-element subsets of T, we can define

$$x < y \iff (\exists w \in y)(\forall v \in x) v < w \tag{1}$$

Defined in this way, $<$ is a strict linear order, but we can't really hope for any more structure than that. But, as promised, with a bit of pruning, we can find a subtree with much more structure.

Theorem 1. *Let T be a finitely branching tree and let $<$ be the lexicographic ordering on the nodes of and paths through T as given above. Define*

$$T' = \{v \in T | \forall w \in [T], v < w\} \tag{2}$$

Then $<$ restricted to T' is order isomorphic to an initial segment of ω. In addition, if T is infinite, T' is infinite as well.

Proof. This is a consequence of the following direct result of König's Lemma.

Lemma 1. *Every element of T' has only finitely many $<$-predecessors.*

Proof. Suppose $v \in T'$ had infinitely many $<$-predecessors. Then we could form the tree,

$$S = \{w \in T' | w < v\} \tag{3}$$

This is, in fact, a tree since T' is a tree and $x \sqsubset y$ implies $x < y$. Since v has infinitely many $<$-predecessors, S is infinite.

By König's Lemma, there must be a path through S, call it P. But S is a subtree of T so $P \in [T]$. By definition of $<$, $P < v$, but $v \in T'$ so $v < P$. This is a contradiction, so v must have only finitely many predecessors. □

Any linear order in which every element has only finitely many predecessors clearly cannot have an infinite descending chain, so must be a well-order. As $\omega + 1$ has an element with infinitely many predecessors, the order type must be an initial segment of ω.

For the second part of Theorem 1, we need an additional lemma.

Lemma 2. *If $[T]$ is non-empty, then $[T]$ has a minimal element in the $<$ ordering.*

Proof. We can inductively construct the minimal element of $[T]$. For any $v \in T$, define $s(v)$ to be the minimal (according to $\leq_v$) child of v that is a member of some element of $[T]$ if such a vertex exists. Note that if there is a path through v, $s(v)$ is defined and there is a path through $s(v)$. If v_0 is the root, then $P = \{s^{(n)}(v_0)\}_{n \in \mathbb{N}}$ is the desired minimal element.

Since $[T]$ is non-empty, there is a path through the root and so, by induction $s^{(n)}(v_0)$ is defined for all n. Therefore P is indeed a path.

To see that P is minimal, let $P \neq P' \in [T]$. Let $v = s^{(m)}(v_0)$ be the largest element (according to $\sqsubseteq$) of $P \cap P'$ and let $w \in P'$ be a child of v. By maximality of v, $w \neq s(v)$ and by construction, $s(v) \leq_v w$. Therefore, $s(v) <_v w$. By the lexicographic ordering, w is greater than all descendants of $s(v)$ and also greater than all ancestors of $s(v)$ (since ancestors of $s(v)$ are also ancestors of w). Therefore, $w > u$ for all $u \in P$ and $P' > P$. □

Now, let T be infinite. Then, by König's Lemma again, $[T]$ is non-empty. Let P be the minimal path in $[T]$ according to Lemma 2. Then $P \subset T'$ since for any $v \in P$ and $P' \in [T]$, $v < P \leq P'$. P is infinite, so T' is infinite as well. □

3.2 Tree of Super-Reduced Words

Recall that we are attempting to develop a canonical way of providing a Turing Machine with a potentially exotic tape with a string as input. Given a tree, the results of the previous section will provide us with a subtree in which the lexicographic ordering is a well-ordering, so in this section we will identify a useful tree within the Cayley graph of a finitely generated group.

We will routinely use the natural correspondence between sequences of generators and group elements given by forming the product of the generators and evaluating in the group. Henceforth, sequences and group elements will be used interchangeably. Of course, multiple sequences will correspond to the same group element, but the sequence should always be clear from context.

Definition 3. *Let G be a group. A* super-reduced word *is a finite sequence of elements of G such that no subword, taken as a product in G is equal to the identity. More precisely, it is a sequence, $g_1, \ldots, g_n$ such that for all $1 \leq i < j \leq n$, $\prod_{k=i}^{j} g_k \neq e$ in G.*

In the context of tape graphs, super-reduced words with symbols from the generating set correspond exactly to non-intersecting finite paths through the tape graph. Note that all prefixes of a super-reduced word are themselves super-reduced.

Form the tree, T, of super-reduced words with symbols from S. Since every group element has at least one super-reduced word corresponding to it and G is infinite, T is a spanning tree for the Cayley graph of G, hence, infinite. Therefore, we can construct an infinite T' as in Section 3.1 where the lexicographic ordering is a well-ordering. We are interested not in T', but on a subtree, which we will call R. To define R we will want another definition.

Definition 4. *We say that two sequences of generators, v and w, are equivalent in G, or $v \equiv_G w$ if they correspond to the same group element. In other words, $v \equiv_G w$ if $v = (s_1, \ldots, s_n)$, $w = (r_1, \ldots, r_m)$ and*

$$\prod_{i=1}^{n} s_i = \prod_{i=1}^{m} r_i \ \ \textit{(in G)} \tag{4}$$

Now we can define R as follows,

$$R = \{v \in T' | (\forall w \in T')\, v \equiv_G w \implies v \leq w\} \tag{5}$$

That is, R is the set of vertices in T' that are lexicographically minimal among sequences that represent the same group element.

It's not obvious at first glance, but R is a tree. Suppose to the contrary that $uv = w \in R$ but $u \notin R$. Then there is some $u' < u$ in R corresponding to the same group element. Since we began with the tree of super-reduced words, u and u' are incomparable in the tree order. Therefore, u and u' must differ at some first location. So, $u'v \equiv_G w$ but uv and $u'v$ differ for the first time at the same location and since $u' < u$, $u'v < uv = w$. This is a contradiction since w was supposed to be minimal among words corresponding to the same group element, so R is indeed a tree.

Also non-obvious is the fact that R is infinite. By the proof of Theorem 1, T' contains a path. As was noted earlier, since we began with the tree of super-reduced words, elements that are comparable in the tree order cannot correspond to the same group element. Therefore, the path in T' corresponds to an infinite collection of group elements. R represents the same group elements since we only pruned redundant representations, so R also represents an infinite number of group elements and is therefore infinite.

Now, for any tape graph, we have a tree that is well-ordered lexicographically, represents infinitely many group elements and represents each individual element at most once. This is how we will provide input. Simply begin with the nth vertex in the well-order containing the nth symbol of the input.

3.3 Power of Turing Machines on Cayley Graphs

One of the first questions to be asked about any new model of computation is whether the class of functions computable by the new model is different from the class of computable functions. For Turing machines on Cayley graphs, this depends rather sensitively on properties of the group producing the tape graph. For example,

Lemma 3. *Let (G, S) be a tape graph. There is a Turing machine over (G, S) that can solve the word problem for G.*

Proof. This process is most easily described with the input on a separate one-dimensional read-only tape. In light of the results of the next section, this is equivalent. Also, we will solve the word problem in the form of determining whether a product of generators is equal to the identity element.

Given a sequence of generators, the head on the tape graph will first mark the origin with a special symbol and then follow the edges given by the sequence on the input tape. If, upon reaching the end of the input, the head on the tape graph is reading the special symbol, accept. Otherwise, reject.

Clearly, this machine only accepts if the head on the tape graph returns to its starting point, which is equivalent to the product of the generators in the input being equivalent.

It has long been known [1,6] that there exist groups with undecidable word problems, so this leads us to believe that Turing machines on a given group with unsolvable word problem are strictly more powerful than standard Turing machines. However, this requires that Turing machines over said group also be able to compute all computable functions. Fortunately, this is the case.

Theorem 2. *Let*

$$M = (Q_M, \Gamma_M, b_M, \Sigma_M, \delta_M, q_{0M}, F_M) \tag{6}$$

be a standard one-dimensional one-way infinite one-tape Turing Machine and let (G, S) be a tape graph. Then there is a Turing Machine over (G, S) that can simulate M.

Here, a standard machine is a one-dimensional machine with a one-way infinite tape. For a technical definition, see Lecture 28 in [5].

The simulation itself is very straight-forward. The only difficulty stems from the question of how to arrange the tape contents of the simulated machine on the Cayley graph. If we could compute an infinite non-self-intersecting path through the Cayley graph, we could use this as a standard one-dimensional tape and do the simulation there. However, such a path need not exist.

Fortunately, we can do the simulation anyway, in this case, by a variant of the "always turn left" algorithm for solving mazes. By putting an ordering on the generators of our tape graph, "always turn left" becomes "always follow the lexicographically minimal edge". Thus, we can do the simulation on the tree constructed in Section 3.2 with the nth vertex in the well-ordering storing the contents of the nth tape cell.

3.4 Proof of Theorem 2

Let (G, S) be the tape graph given in the statement of the theorem. Denote the elements of S by $g_1, \ldots, g_n$. It will be convenient to define a set $S' = S \cup \{g_0, g_{n+1}\}$ with the ordering $g_0 < g_1 < \cdots < g_n < g_{n+1}$. We will consider both g_0 and g_{n+1} equal to e in the group for the purposes of moving from node to node. Define a Turing Machine, N, over (G, S) as follows (here, $\sqcup$ refers to a disjoint union):

- $Q_N = Q_M \sqcup (Q_M \times S') \sqcup (Q_M \times S') \sqcup (Q_M \times S) \sqcup (Q_M \times S)$
- $\Gamma_N = \Gamma_M \times S' \times \mathcal{P}(S) \times \mathcal{P}(S)$
- $b_N = (b_M, g_0, \emptyset, \emptyset)$
- $\Sigma_N = \Sigma_M \times S' \times \mathcal{P}(S) \times \mathcal{P}(S)$
- $q_{0N} = q_{0M}$
- $F_N = F_M \times S' \times \mathcal{P}(S) \times \mathcal{P}(S)$

Each state in the tape alphabet will have an intended meaning. Remember that we are going to do the computation on a tree, so we have to encode in each node not just the symbol of the simulated machine, but also auxiliary information about the structure of the tree. In particular, if $(\gamma, \sigma, A, B) \in \Gamma_N$, γ is the symbol of the simulated machine stored at the node, σ is the generator to follow to reach the ancestor of this node in the tree, A is the set of generators corresponding to edges pointing away from the root in the tree, and B is the set of generators defining non-edges of the tree. Remember that we are going to be constructing this tree on the fly and so we don't have complete information about which generators correspond to edges of the tree at every step. Thus, elements of $S \setminus (A \cup B)$ are the edges whose membership in the tree has not yet been determined.

We will also give each state a name and an intended interpretation,

- Cq for $q \in Q_M$: We are simulating the computation of M and the current state of M is q.
- Rqx for $q \in Q_M$ and $x \in S'$: The simulated tape head is moving to the right and M is currently in state q. The argument x encodes the edge we followed to reach our current location.

- Lqx for $q \in Q_M$ and $x \in S'$: The simulated tape head is moving to the left and M is currently in state q. The argument x encodes the edge we followed to reach our current location.
- Eqx for $q \in Q_M$ and $x \in S$: The simulated tape head is moving to the right and M is in state q, but we have run out of tape and are attempting to extend the tree along edge x.
- Bqx for $q \in Q_M$ and $x \in S$: M is currently in state q and we just failed to extend the tree along edge x, so we are backtracking.

Note that the starting state of N is named Cq_{0M}.

It will also be convenient to talk about the component functions of the transition function of M,

$$\begin{aligned} \delta_1 &: Q_M \times \Gamma_M \to Q_M \\ \delta_2 &: Q_M \times \Gamma_M \to \Gamma_M \\ \delta_3 &: Q_M \times \Gamma_M \to \{L, R\} \end{aligned}$$

We can now write down the action of the transition function. If the current symbol being read is (γ, σ, A, B), then the value of the transition function on each type of state is given in Table 1. Entries in the table are triples in the order, new state, symbol written, direction the tape head moves.

Table 1. Transition Table for N

$\digamma \in \Gamma_N$	$\delta_N(\digamma, (\gamma, \sigma, A, B))$
Cq	$\begin{cases} (R\delta_1(q,\gamma)g_0, (\delta_2(q,\gamma), \sigma, A, B), g_0) & \text{if } \delta_3(q,\gamma) = R \\ (L\delta_1(q,\gamma)\sigma, (\delta_2(q,\gamma), \sigma, A, B), \sigma) & \text{if } \delta_3(q,\gamma) = L \end{cases}$
Lqx	$\begin{cases} (Cq, (\gamma, \sigma, A, B), g_0) & \text{if } \forall y \in A, y \geq x \\ (Lqg_{n+1}, (\gamma, \sigma, A, B), \max_{y \in A, y < x} y) & \text{otherwise} \end{cases}$
Rqx	$\begin{cases} (Cq, (\gamma, \sigma, A, B), \min_{y \in A, y > x} y) & \text{if } \exists y \in A, y > x \\ (Eqy, (\gamma, \sigma, A \cup \{y\}, B), y) & \text{where } y = \min_{z \in S \setminus B, z > x} z \\ & \text{if } \exists z \in S \setminus B, z > x \\ (Rq\sigma, (\gamma, \sigma, A, B), \sigma) & \text{otherwise} \end{cases}$
Eqx	$\begin{cases} (Cq, (\gamma, x^{-1}, \emptyset, \{x^{-1}\}), g_0) & \text{if } A = B = \emptyset \\ (Bqy, (\gamma, \sigma, A, B), x^{-1}) & \text{otherwise} \end{cases}$
Bqx	$(Rqx, (\gamma, \sigma, A \setminus \{x\}, B \cup \{x\}), g_0)$

Most of these transitions are pretty opaque, so some explanation is warranted. If we are in a C-state, then we perform one step of the computation and then transition into either an R-state or an L-state depending on whether the computation tries to move left or right. A leftward step in the simulated machine always begins with a step toward the root for the simulating machine, so we take that step immediately. Rightward steps are more complicated, so we leave the tape head where it is on a rightward step.

Taking a step to the left, we want to find the lexicographic immediate predecessor of our current vertex. We begin by taking one step toward the root, which we did when we transitioned out of the C-state. Then, either there is a branch all of whose elements are less than where we started, or not. If not, then we are at the immediate predecessor of our origin, and we continue computing. Otherwise, follow the greatest such branch and always move to the greatest child until we reach a dead-end. This dead-end is the immediate predecessor we were looking for, so we can continue the computation.

Taking a step to the right is significantly more complicated, as we are most likely going to have to extend the tree as we go. If we have already built some tree above us (A is non-empty), then we simply use that part of the tree, moving along the minimal edge in the tree and continuing the computation. This is the reason for using the "false" generator g_0 when we transition out of the C-state. Otherwise, try to extend the tree along the least edge that we have not already ruled out. We add that edge to A and switch to an E-state.

If A and B are both empty, then this is a new vertex that we have not visited before, so continue with the computation. Otherwise, we are at a vertex that has already been added elsewhere in the tree. So, we back up and transition to a B-state. In the B-state, we rule out the edge we just took by removing it from A and adding it to B and switch to an R-state to try extending the tree again.

If we can't extend the tree at all ($B = \{g_1, \ldots, g_n\}$), then take a step toward the root and try again, eschewing anything less than or equal to the edge we backtracked along. Since there is an infinite tree for us to use, we will eventually be able to extend the tree and continue the computation.

3.5 An Alternative Characterization of the r.e. Turing Degrees

We have demonstrated that Turing Machines on arbitrary Cayley graphs are strictly more powerful than standard Turing Machines, so the next question to ask is, "how much more powerful?" The short answer is "as powerful as we want". In [2,3] Boone showed that for any r.e. Turing degree, there is a finitely presented group whose word problem is in that degree. Using such a group, we can produce a machine (more precisely a class of machines) that computes exactly the functions in or below the given degree. More precisely,

Theorem 3. *Let T be an r.e. Turing degree. There is a group, G, such that the class of functions computable by a Turing Machine over G is exactly the class of functions in or below T (with respect to Turing reducibility).*

Proof. By Boone, let G be a group whose word problem is in T and let f be a function in or below T. Since $f \leq_T T$, f is Turing reducible to the word problem for G. Turing machines over G can perform this reduction since they can compute all recursive functions and they can solve the word problem for G by Lemma 3. Therefore, they can compute f.

To show that a Turing Machine over G cannot compute a class larger than T, observe that a Turing Machine with an oracle for the word problem for G can easily simulate a Turing Machine over G. It simply maintains a list of nodes written as a sequence of generators for the address together with whichever tape symbol is written there. When the simulated tape head moves, the machine simply appends the generator to the current address and consults the oracle to determine which node this corresponds to, adding a new entry if the new node isn't in the list. □

3.6 Further Remarks

As was previously mentioned, generalizing to multiple tapes and multiple heads in this context is done in the same way as in the context of standard Turing machines. Many of the simulation results carry over directly as well. For example, a one-tape machine with multiple heads can be simulated by a one-tape machine with one head. The locations of all the heads can simply be marked on the tape with special symbols and for each step of the simulated machine, the simulating machine can search through the entire tape for all the head symbols and then update the tape accordingly. Similarly, a machine with multiple, identical tapes can be simulated by a machine with a single tape of the same type. Simply enlarge the tape alphabet to tuples of tape symbols together with special symbols for the head on each tape and follow the procedure for multiple heads.

The only major difference is in a machine with multiple, different tapes. In this case, the class of functions computable by the machine is the class of functions in or below the join of the r.e. degrees of the word problems of the tapes. Just as in Section 3.5, this machine is mutually simulatable with a standard Turing machine with oracles for the word problem of each tape. In fact, this case subsumes the multiple-head, single-tape and multiple-head, multiple-tape cases.

Acknowledgements. The author would like to thank Andreas Blass for comments on early versions and the anonymous referees who provided both useful comments and historical background.

References

1. Boone, W.W.: The word problem. Proc. Nat. Acad. Sci. U.S.A. 44, 1061–1065 (1958)
2. Boone, W.W.: Word problems and recursively enumerable degrees of unsolvability. A first paper on Thue systems. Ann. of Math. 83(2), 520–571 (1966)
3. Boone, W.W.: Word problems and recursively enumerable degrees of unsolvability. A sequel on finitely presented groups. Ann. of Math. 84(2), 49–84 (1966)
4. Grigorév, D.J.: Time complexity of multidimensional Turing machines. Zap. Nauchn. Sem. Leningrad. Otdel. Mat. Inst. Steklov. (LOMI) 88, 47–55 (1979); Studies in constructive mathematics and mathematical logic, VIII
5. Kozen, D.: Automata and Computability. Springer, New York (1997)
6. Novikov, P.S.: On algorithmic unsolvability of the problem of identity. Dokl. Akad. Nauk. SSSR 85, 709–719 (1952)

Information Flow on Directed Acyclic Graphs

Michael Donders, Sara Miner More, and Pavel Naumov

Department of Mathematics and Computer Science,
McDaniel College, Westminster, Maryland 21157, USA
{msd002,smore,pnaumov}@mcdaniel.edu

Abstract. The paper considers a multi-argument *independence* relation between messages sent over the edges of a directed acyclic graph. This relation is a generalization of a relation known in information flow as nondeducibility. A logical system that describes the properties of this relation for an arbitrary fixed directed acyclic graph is introduced and proven to be complete and decidable.

1 Introduction

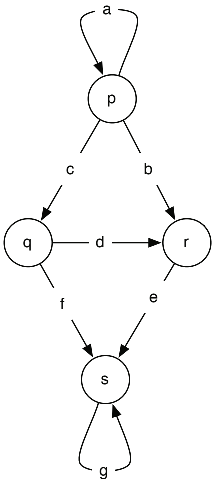


Fig. 1. Graph G_0

In this paper we study information flow on directed acyclic graphs. We view directed graphs as communication networks in which vertices are parties and directed edges are communication channels. An example of such a graph, G_0, is depicted in Figure 1. We use loop edges to represent values that are computed by the party, but not sent to anyone else. The conditions that parties must observe while communicating over the network will be called *action relations.* The set of action relations for all vertices will be called a *protocol.* Here is a sample protocol $\mathcal{P}_0$ over graph G_0: vertex p picks a random boolean value $a \in \{0,1\}$ and finds two boolean values c and b such that $a \equiv b + c \pmod 2$. It sends value c to vertex q and value b to vertex r. Vertex q finds boolean values d and f such that $c \equiv d + f \pmod 2$ and sends them to vertices r and s, respectively. Vertex r computes the value $e \equiv d + b \pmod 2$ and sends it to vertex s. Vertex s computes value $g \equiv f + e \pmod 2$.

An assignment of values to all channels that satisfies all action relations will be called a *run* of the protocol. Note that for the protocol described above, values c and b are independent in the sense that any possible value of c may occur on the same run with any possible value of b. We denote this by $[c, b]$. This relation between two values was originally introduced by Sutherland [1] and later became known in the study of information flow as *nondeducibility.* Halpern and O'Neill [2] proposed a closely-related notion called f-secrecy. More and Naumov [3] generalized nondeducibility to a relation between an arbitrary set of values and called it independence. For example, values c, b, and d for the above protocol are independent in the sense that any

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 95–109, 2011.

combination of their possible values may occur on the same run. We denote this relation by $[c, b, d]$. At the same time, it is easy to see that under the above protocol:

$$g \equiv f + e \equiv f + (d + b) \equiv (f + d) + b \equiv c + b \equiv a \pmod 2. \tag{1}$$

Thus, not every combination of values of a and g can occur together on a run. In our notation: $\neg[a, g]$.

The properties mentioned above are specific to the given protocol. If the protocol changes, some of the true properties can become false and vice versa. In this paper, however, we focus on the properties that are true for a given graph no matter which protocol is used. An example of such property for the above graph is $[c, b, f, e] \to [a, g]$. It says that for any protocol under G_0 if values c, b, f and e are independent over this protocol, then values a and g are also independent under the same protocol. We will formally prove this claim in Proposition 3.

The main result of this paper is a sound and complete logical system that describes all propositional properties of the multi-argument relation $[a_1, \ldots, a_n]$ on directed graphs which are acyclic, with the possible exception of loop edges. Previously, More and Naumov obtained similar results for undirected graphs [3] and hypergraphs [4]. Compared to the case of undirected graphs, the logical system described here adds an additional Directed Truncation inference rule.

Our logical system describes information flow properties of a graph, not a specific protocol over this graph. However, this system can be used to reason about the properties of a specific protocol by treating some properties of the protocol as axioms, then using our system to derive additional properties of the protocol.

2 Protocol: A Formal Definition

Throughout this work, by a graph we mean a finite directed graph with cycles of length no more than one or, less formally, "directed acyclic graphs with loops". Such graphs define a partial order on vertices that will be assumed to be the order in which the protocol is executed. The protocol will specify how the values on outgoing edges are related to the values one the incoming edges of each vertex. With this in mind, we will count loops at any vertex v among its outgoing edges $Out(v)$, but *not* among its incoming edges $In(v)$.

Definition 1. *A protocol over a graph $G = \langle V, E\rangle$ is a pair $\langle M, \Delta\rangle$ such that*

1. *$M(e)$ is an arbitrary set of values ("messages") for each edge $e \in E$,*
2. *$\Delta = \{\Delta_v\}_{v\in V}$ is a family of action relations between values of incoming and outgoing edges of the vertex v:*

$$\Delta_v \subseteq \left(\prod_{e\in In(v)} M(e)\right) \times \left(\prod_{e\in Out(v)} M(e)\right).$$

3. **(continuity condition)** *For any possible tuple of values on the incoming edges of a vertex v, there is at least one tuple of values possible on its outgoing edges:*

$$\forall x \in \prod_{e \in In(v)} M(e) \quad \exists y \in \prod_{e \in Out(v)} M(e) \Bigg((x, y) \in \Delta_v \Bigg).$$

The continuity condition above distinguishes protocols over directed graphs from protocols over undirected graphs [3].

Definition 2. *A run of a protocol $\mathcal{P} = \langle M, \Delta \rangle$ over graph $G = \langle V, E \rangle$ is any function r on E such that*

1. *$r(e) \in M(e)$ for each $e \in E$,*
2. *$\langle \langle r(e) \rangle_{e \in In(v)}, \langle r(e) \rangle_{e \in Out(v)} \rangle \in \Delta_v$ for each $v \in V$.*

The set of runs of a protocol $\mathcal{P}$ is denoted by $\mathcal{R}(\mathcal{P})$.

Definition 3. *A protocol $\mathcal{P} = \langle M, \Delta \rangle$ over graph $G = \langle E, V \rangle$ is called finite if the set $M(e)$ is finite for each edge $e \in E$.*

We conclude with the definition of a multi-argument version of Sutherland's binary nondeducibility predicate called *independence*. It is identical to the one used by More and Naumov [3,4].

Definition 4. *A set of edges A is called independent under protocol $\mathcal{P}$ if for any family of runs $\{r_a\}_{a \in A} \subseteq \mathcal{R}(\mathcal{P})$ there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r(a) = r_a(a)$ for each $a \in A$.*

In the above definition, we refer to the value $r_a(a)$, rather than an arbitrary element of $M(a)$, because there may be some values in $M(a)$ that are not actually used on any given run. In the next section, we will formally define the formulas of our logic system. The atomic formula expressing the independence of a set A will be denoted by $[A]$.

3 Semantics

Informally, by $\Phi(G)$ we denote the set of all propositional properties of independence over a fixed graph $G = \langle V, E \rangle$. Formally, $\Phi(G)$ is a minimal set defined recursively as follows: (i) for any finite set of edges $A \subseteq E$, formula $[A]$ belongs to set $\Phi(G)$, (ii) the false constant $\bot$ belongs to $\Phi(G)$, and (iii) for any formulas ϕ and ψ in $\Phi(G)$, the implication $\phi \to \psi$ also belongs to $\Phi(G)$. Conjunction, disjunction, and negation will be assumed to be defined through connectives $\to$ and $\bot$.

Next, we define the relation $\mathcal{P} \vDash \phi$ between a protocol $\mathcal{P}$ over graph G and a formula $\phi \in \Phi(G)$. Informally, $\mathcal{P} \vDash \phi$ means that formula ϕ is true under $\mathcal{P}$.

Definition 5. *For any protocol $\mathcal{P}$ over a graph G, and any formula $\phi \in \Phi(G)$, we define the relation $\mathcal{P} \vDash \phi$ recursively as follows: (i) $\mathcal{P} \nvDash \bot$, (ii) $\mathcal{P} \vDash [A]$ if the set of edges A is independent under protocol $\mathcal{P}$, (iii) $\mathcal{P} \vDash \phi_1 \to \phi_2$ if $\mathcal{P} \nvDash \phi_1$ or $\mathcal{P} \vDash \phi_2$.*

We will illustrate this definition with the two propositions below. By G_0 we mean the graph depicted earlier in Figure 1.

Proposition 1. *There is a protocol $\mathcal{P}$ over G_0 such that $\mathcal{P} \nvDash [b, f, g] \to [a, g]$.*

Proof. Consider the protocol $\mathcal{P}$ under which the party (represented by vertex) p picks a boolean value a and sends it via edge c to party q . In other words, $a = c$ is the action relation at vertex p. At the same time, the constant value 0 is sent via edge b, which means that $M(b) = \{0\}$. Party q resends value c through edge d and sends the constant 0 through edge f. Party r then resends value d through edge e and, finally, s resends value e through channel g. Under this protocol, $M(b) = M(f) = \{0\}$. Thus, any possible values of edges b, f, and g may occur on the same run. In other words, $\mathcal{P} \vDash [b, f, g]$. At the same time, $a = c = d = e = g$, and $M(a) = M(g) = \{0, 1\}$. Thus, not every combination of values of a and g can occur on the same run. Therefore, $\mathcal{P} \nvDash [a, g]$. □

Note that in the proof of the previous proposition direction of edge d is important. One might expect that the result is not true if the direction of the edge d is reversed. This, however, is not true:

Proposition 2. *There is a protocol $\mathcal{P}$ over G_0 such that $\mathcal{P} \nvDash [c, e, g] \to [a, g]$.*

Proof. Consider the protocol $\mathcal{P}_0$ over G_0 described in the introduction. It was shown earlier through equality (1), that $\mathcal{P}_0 \nvDash [a, g]$. Thus, we only need to prove that $\mathcal{P}_0 \vDash [c, e, g]$. Let c_0, e_0, g_0 be any boolean values. We will show that these values can co-exist on the same run. Indeed, let $f_0 = e_0 + g_0 \pmod 2$, $d_0 = c_0 + f_0 \pmod 2$, $b_0 = d_0 + e_0 \pmod 2$, and $a_0 = c_0 + b_0 \pmod 2$. It is easy to see that values $a_0, b_0, c_0, d_0, e_0, f_0$, and g_0 form a valid run of $\mathcal{P}_0$. □

In this paper, we study the set of formulas that are true under *any* protocol $\mathcal{P}$ as long as the graph G remains fixed. The set of all such formulas will be captured by our logical system for information flow over directed acyclic graphs. This system is described in Section 5.

4 Graph Notation

Before the introduction of our formal system, we need to define some graph-related notation that will be used in this system.

A *cut* of a graph is a disjoint partitioning of its vertices into two sets. A *crossing edge* of a cut is an edge whose ends belong to different sets of the partition. For any set of vertices X of a graph G, we use $E(X)$ to denote the set of all edges of G whose ends both belong to X.

Definition 6. *Let G be an arbitrary graph and (X, Y) be an arbitrary cut of G. We define the "truncation" graph G_X of graph G as follows:*

1. *The vertices of graph G_X are the vertices of set X.*
2. *The edges of G_X are all of the edges from $E(X)$ plus the crossing edges of the cut (X, Y) modified in the following way: if, in graph G, a crossing edge c connects vertex $v \in X$ with a vertex in Y, then, in graph G_X, edge c loops from v back into v.*

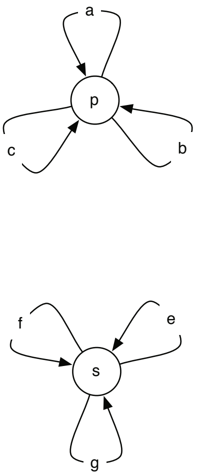


Fig. 2. Graph G'_0

Each edge e in a truncated graph G_X corresponds to a unique edge in the original graph G. Although the two corresponding edges might connect different vertices in their respective graphs, we will refer to both of them as edge e. For example, graph G'_0 in Figure 2 is obtained from graph G_0 in Figure 1 by truncating along the cut $(\{p, s\}, \{q, r\})$. In the above notation, this truncated graph can be denoted by $(G_0)_{\{p,s\}}$.

Definition 7. *A cut (X, Y) is called "directed" if there are no crossing edges of this cut that lead from Y to X.*

Definition 8. *A gateway between sets of edges A and B in a graph G is a set of edges W such that every undirected path from A to B contains at least one edge from W.*

Note that sets A, B, and W are not necessarily disjoint. Thus, for example, for any set of edges A, set A is a gateway between A and itself. Also, note that the empty set is a gateway between any two components of the graph that are not connected one to another.

5 Formal System: Axioms and Rules

We are now ready to describe our logical system for information flow over directed acyclic graphs. We will write $G \vdash \phi$ to state that formula $\phi \in \Phi(G)$ is provable in this logic. Everywhere below, X, Y denotes the union of sets X and Y. In addition to all propositional tautologies and the Modus Ponens inference rule, the deductive system for this logic consists of the *Small Set* axiom, the *Gateway* axiom, and the *Truncation* and the *Directed Truncation* inference rules:

Small Set Axiom. Any set that contains less than two edges is independent: $G \vdash [A]$, where $A \subseteq E$ and $|A| < 2$.

Gateway Axiom. $G \vdash [A, W] \rightarrow ([B] \rightarrow [A, B])$, where W is a gateway between sets of edges A and B such that $A \cap W = \varnothing$.

Truncation Rule. Let C be the set of all crossing edges of a cut (X, Y) and ϕ be a formula in $\Phi(G_X)$. If $G_X \vdash \phi$, then $G \vdash [C] \rightarrow \phi$.

Directed Truncation Rule. Let (X, Y) be a directed cut and $\phi \in \Phi(G_X)$. If $G_X \vdash \phi$, then $G \vdash \phi$.

The soundness of this system will be demonstrated in Section 6 and its completeness in Section 7. Below, we present a general result to which we will refer during the proof of completeness.

Theorem 1 (monotonicity). $G \vdash [A] \to [B]$*, for any graph G and any subsets* $B \subseteq A$ *of edges of* G*.*

Proof. Consider sets B and $\varnothing$. Since there are no paths connecting these sets, any set of edges is a gateway between these sets. In particular $(A \setminus B)$ is such a gateway. Taking into account that sets B and $(A \setminus B)$ are disjoint, by the Gateway axiom, $G \vdash [B, (A \setminus B)] \to ([\varnothing] \to [B])$. By the Small Set axiom, $G \vdash [\varnothing]$. Thus, $G \vdash [B, (A \setminus B)] \to [B]$. By the assumption $B \subseteq A$, we conclude that $G \vdash [A] \to [B]$. □

Next we give two examples of derivations in our logical system. In these examples, by G_0 we mean the graph depicted earlier in Figure 1.

Proposition 3. $G_0 \vdash [c, b, f, e] \to [a, g]$.

Proof. We will start with graph G_0' depicted in Figure 2. Recall that this graph is obtained from G_0 by truncation with crossing edges c, b, f and e. Note that, in graph G_0', the empty set is a gateway between sets $\{a\}$ and $\{g\}$. Thus, by the Gateway axiom, $G_0' \vdash [a] \to ([g] \to [a, g])$. By the Small Set axiom, $G_0' \vdash [a]$ and $G_0' \vdash [g]$. Hence, $G_0' \vdash [a, g]$. By the Truncation rule, $G_0 \vdash [c, b, f, e] \to [a, g]$. □

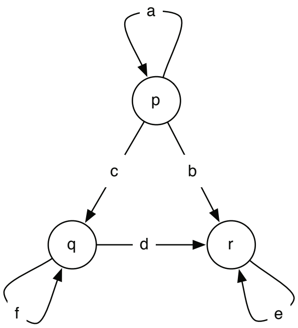


Fig. 3. Graph G_0''

Proposition 4. $G_0 \vdash [c, b, d] \to [c, e]$.

Proof. Consider graph G_0'' depicted in Figure 3. It is obtained from graph G by a truncation with crossing edges e and f. Note that in graph G_0'' set $\{b, d\}$ is a gateway between sets $\{c\}$ and $\{e\}$. Thus, by the Gateway axiom, $G_0'' \vdash [c, b, d] \to ([e] \to [c, e])$. By the Small Set axiom, $G_0'' \vdash [e]$. Hence, $G_0'' \vdash [c, b, d] \to [c, e]$. By the Directed Truncation rule, $G_0 \vdash [c, b, d] \to [c, e]$. □

6 Soundness

The proof of soundness is non-trivial. For each axiom and inference rule, we provide its justification as a separate theorem.

Theorem 2 (Small Set). *For any graph* $G = \langle V, E\rangle$*, if* $\mathcal{P}$ *is an arbitrary protocol over* G *and subset* $A \subseteq E$ *has at most one element, then* $\mathcal{P} \vDash [A]$*.*

Proof. Case 1: $A = \varnothing$. Due to the continuity condition in Definition 1 and because graph G is acyclic, there is at least one run $r \in \mathcal{R}(\mathcal{P})$. Thus, $\mathcal{P} \vDash [\varnothing]$. *Case 2:* $A = \{a_1\}$. Consider any run $r_1 \in \mathcal{R}(\mathcal{P})$. Pick r to be r_1. This guarantees that $r(a_1) = r_1(a_1)$. □

Theorem 3 (Gateway). *For any graph $G = \langle V, E\rangle$, and any gateway W between sets of edges A and B in graph G, if $\mathcal{P} \vDash [A, W]$, $\mathcal{P} \vDash [B]$, and $A \cap W = \varnothing$, then $\mathcal{P} \vDash [A, B]$.*

Proof. Assume $\mathcal{P} \vDash [A, W]$, $\mathcal{P} \vDash [B]$, and $A \cap W = \varnothing$. Let $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_k\}$. Consider any $r_1, \dots, r_{n+k}$. We will show that there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r(a_i) = r_i(a_i)$ for each $i \leq n$ and $r(b_i) = r_{n+i}(b_i)$ for each $i \leq k$. By the assumption $\mathcal{P} \vDash [B]$, there is a run $r_B \in \mathcal{R}(\mathcal{P})$ such that

$$r_B(b_i) = r_{n+i}(b_i) \qquad \text{for } i \leq k. \tag{2}$$

By assumptions $\mathcal{P} \vDash [A, W]$ and $A \cap W = \varnothing$, there must be a run r_A such that

$$r_A(e) = \begin{cases} r_i(e) & \text{if } e = a_i \text{ for } i \leq n, \\ r_B(e) & \text{if } e \in W. \end{cases} \tag{3}$$

Next, consider graph G' obtained from G by removing all edges in W. By the definition of gateway, no single connected component of graph G' can contain both an edge from A and an edge from $(B \setminus W)$. Let us group all connected components of G' into two subgraphs G'_A and G'_B such that G'_A contains no edges from $(B \setminus W)$ and G'_B contains no edges from A. Components that contain edges neither from A nor from $(B \setminus W)$ can be arbitrarily assigned to either G'_A or G'_B.

By equation (3), runs r_A and r_B on G agree on each edge of gateway W. We will now construct a combined run r by "sewing together" portions of r_A and r_B with the "stitches" placed along gateway W. Formally,

$$r(e) = \begin{cases} r_A(e) & \text{if } e \in G'_A, \\ r_A(e) = r_B(e) & \text{if } e \in W, \\ r_B(e) & \text{if } e \in G'_B. \end{cases} \tag{4}$$

Let us first prove that r is a valid run of the protocol $\mathcal{P}$. For this, we need to prove that it satisfies action relation Δ_v at every vertex v. Without loss of generality, assume that $v \in G'_A$. Hence, on all edges incident with v, run r agrees with run r_A. Thus, run r satisfies Δ_v simply because r_A does.

Next, we will show that $r(a_i) = r_i(a_i)$ for each $i \leq n$. Indeed, by equations (3) and (4), $r(a_i) = r_A(a_i) = r_i(a_i)$. Finally, we need to show that $r(b_i) = r_{n+i}(b_i)$ for each $i \leq k$. This, however, follows easily from equations (2) and (4). □

Theorem 4 (Truncation). *Assume that C is the set of all crossing edges of cut (X, Y) in graph G and ϕ is a formula in $\Phi(G_X)$. If $\mathcal{P}' \vDash \phi$ for each protocol $\mathcal{P}'$ over G_X, then $\mathcal{P} \vDash [C] \to \phi$ for each protocol $\mathcal{P}$ over graph G.*

Proof. Suppose that there is a protocol $\mathcal{P}$ over G such that $\mathcal{P} \vDash [C]$, but $\mathcal{P} \nvDash \phi$. We will construct a protocol $\mathcal{P}'$ over G_X such that $\mathcal{P}' \nvDash \phi$.

Let $\mathcal{P} = \langle M, \Delta\rangle$. Note that, for any edge e, not all values from $M(e)$ are necessarily used in the runs of this protocol. Some values might be excluded by the action relations of $\mathcal{P}$. To construct protocol $\mathcal{P}' = \langle M', \Delta'\rangle$ over truncation

G_X, for any edge e of G_X we first define $M'(e)$ as the set of values that are actually used by at least one run of protocol $\mathcal{P}$. Thus, $M'(e) = \{r(e) \mid r \in \mathcal{R}(\mathcal{P})\}$. The action relation Δ'_v at any vertex v of G_X is the same as under protocol $\mathcal{P}$.

Lemma 1. *For any run $r' \in \mathcal{R}(\mathcal{P}')$ there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r(e) = r'(e)$ for each edge e in truncation G_X.*

Proof. Consider any run $r' \in \mathcal{R}(\mathcal{P}')$. By the definition of M', for any crossing edge $c \in C$, there is a run $r_c \in \mathcal{R}(\mathcal{P})$ such that $r'(c) = r_c(c)$. Since $\mathcal{P} \vDash [C]$, there is a run $r_Y \in \mathcal{R}(\mathcal{P})$ such that $r_Y(c) = r_c(c) = r'(c)$ for each $c \in C$.

We will now construct a combined run $r \in \mathcal{R}(\mathcal{P})$ by "sewing together" r_Y and r' with the "stitches" placed in set C. Recall that we use the notation $E(X)$ to denote edges whose ends are both in set X. Formally, let

$$r(e) = \begin{cases} r'(e) & \text{if } e \in E(X), \\ r'(e) = r_Y(e) & \text{if } e \in C, \\ r_Y(e) & \text{if } e \in E(Y). \end{cases}$$

We just need to show that r satisfies Δ_v at every vertex v of graph G. Indeed, if $v \in Y$, then run r is equal to r_Y on all edges incident with v. Thus, it satisfies the action relation at v because run r_Y does. Alternatively, if $v \in X$, then run r is equal to run r' on all edges incident with v. Since r' satisfies action relation Δ'_v and, by definition, $\Delta'_v \equiv \Delta_v$ for all $v \in X$, we can conclude that r again satisfies condition Δ_v. □

Lemma 2. *For any set of edges Q in graph G_X, $\mathcal{P} \vDash [Q]$ if and only if $\mathcal{P}' \vDash [Q]$.*

Proof. Assume first that $\mathcal{P} \vDash [Q]$ and consider any runs $\{r'_q\}_{q \in Q} \subseteq \mathcal{R}(\mathcal{P}')$. We will construct a run $r' \in \mathcal{R}(\mathcal{P}')$ such that $r'(q) = r'_q(q)$ for every $q \in Q$. Indeed, by Lemma 1, there are runs $\{r_q\}_{q \in Q} \subseteq \mathcal{R}(\mathcal{P})$ that match runs $\{r'_q\}_{q \in Q}$ on all edges in G_X. By the assumption that $\mathcal{P} \vDash [Q]$, there must be a run $r \in \mathcal{R}(\mathcal{P})$ such that $r(q) = r_q(q)$ for all $q \in Q$. Hence, $r(q) = r_q(q) = r'_q(q)$ for all $q \in Q$. Let r' be the restriction of run r to the edges in G_X. Since the action relations of protocols $\mathcal{P}$ and $\mathcal{P}'$ are the same at all vertices in X, we can conclude that $r' \in \mathcal{R}(\mathcal{P}')$. Finally, we notice that $r'(q) = r(q) = r'_q(q)$ for any $q \in Q$.

Next, assume that $\mathcal{P}' \vDash [Q]$ and consider any runs $\{r_q\}_{q \in Q} \subseteq \mathcal{R}(\mathcal{P})$. We will show that there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r(q) = r_q(q)$ for all $q \in Q$. Indeed, let $\{r'_q\}_{q \in Q}$ be the restrictions of runs $\{r_q\}_{q \in Q}$ to the edges in G_X. Since the action relations of these two protocols are the same at the vertices in X, we can conclude that $\{r'_q\}_{q \in Q} \subseteq \mathcal{R}(\mathcal{P}')$. By the assumption that $\mathcal{P}' \vDash [Q]$, there is a run $r' \in \mathcal{R}(\mathcal{P}')$ such that $r'(q) = r'_q(q) = r_q(q)$ for all $q \in Q$. By Lemma 1, there is a run $r \in \mathcal{R}(\mathcal{P})$ that matches r' everywhere in G_X. Therefore, $r(q) = r'(q) = r_q(q)$ for all $q \in Q$. □

Lemma 3. *For any formula $\psi \in \Phi(G_X)$, $\mathcal{P} \vDash \psi$ if and only if $\mathcal{P}' \vDash \psi$.*

Proof. We use induction on the complexity of ψ. The base case follows from Lemma 2, and the induction step is trivial. □

The statement of Theorem 4 immediately follows from Lemma 3. □

Theorem 5 (Directed Truncation). *Assume that* (X, Y) *is a directed cut of a graph* G *and* ϕ *is a formula in* $\Phi(G_X)$*. If* $\mathcal{P}' \vDash \phi$ *for every protocol* $\mathcal{P}'$ *over truncation* G_X*, then* $\mathcal{P} \vDash \phi$ *for every protocol* $\mathcal{P}$ *over graph* G*.*

The proof of this theorem is a straightforward modification of the proof of Theorem 4. Specifically, in the proof of Lemma 1, instead of "sewing together" runs r' and r_Y, we use the continuity condition from Definition 1 to extend run $r' \in \mathcal{R}(\mathcal{P}')$ into a run $r \in \mathcal{R}(\mathcal{P})$ that agrees with r' on all vertices in G_X.

7 Completeness

Theorem 6 (completeness). *For any directed graph* G*, if* $\mathcal{P} \vDash \phi$ *for all finite protocols* $\mathcal{P}$ *over* G*, then* $G \vdash \phi$*.*

The theorem will be proven by contrapositive. At the core of this proof is the construction of a finite protocol. This protocol will be formed as a composition of several simpler protocols, where each of the simpler protocols is defined recursively. The base case of this recursive definition is the parity protocol defined below. It is a generalization of the protocol described in the introduction.

7.1 Parity Protocol

In the following discussion, we use the overloaded notation $Inc(x)$ to denote the set of objects incident with an object x in a graph, where x may be either an edge or a vertex. That is, if x is an edge, then $Inc(x)$ represents the set of (at most two) vertices which are the ends of edge x. On the other hand, if x is a vertex, then $Inc(x)$ represents the set of edges which have vertex x as an end.

Let $G = \langle V, E \rangle$ be a graph and A be a subset of E. We define the "parity protocol" $\mathcal{P}_A$ over G as follows. The set of values of any edge e in graph G is the set of boolean functions on the ends of e (each loop edge is assumed to have a single end). Thus, a run r of the protocol will be a function that maps an edge into a function from the ends of this edge into boolean values: $r(e)(v) \in \{0, 1\}$, where e is an edge and v is an end of e. It will be more convenient, however, to think about a run as a two-argument function $r(e, v) \in \{0, 1\}$.

Not all assignments of boolean values to the ends of an edge e will be permitted in the parity protocol. Namely, if $e \notin A$, then the sum of all values assigned to the ends of e must be even. This is formally captured by the following condition:

$$\sum_{v \in Inc(e)} r(e, v) \equiv 0 \pmod{2}. \tag{5}$$

This means that if an edge $e \notin A$ has two ends, then the values assigned to its two ends must be equal. If edge $e \notin A$ is a loop edge and, thus, has only one end, then the value assigned to this end must be 0. However, if $e \in A$, then no restriction on the assignment of boolean values to the ends of e will be imposed. This defines the set of values $M(e)$ for each edge e under $\mathcal{P}_A$.

The second restriction on the runs will require that the sum of all values assigned to ends incident with any vertex v is also even:

$$\sum_{e \in Inc(v)} r(e,v) \equiv 0 \pmod 2. \tag{6}$$

The latter restriction specifies the action relation Δ_v for each vertex v. We will graphically represent a run by placing boolean values at each end of each edge of the graph. For example, Figure 4 depicts a possible run of the parity protocol $\mathcal{P}_A$ with $A = \{c, b, g\}$ over the graph G_0 from Figure 1.

The finite protocol $\mathcal{P}_A$ is now completely defined, but we still need to prove that it satisfies the continuity condition from Definition 1. This is true, however, only under an additional assumption:

Lemma 4. *If set A is such that it contains a loop edge for each sink of graph G, then $\mathcal{P}_A$ satisfies the continuity condition.*

Proof. As long as a vertex has at least one outgoing edge whose boolean value is not fixed, this value an be adjusted to satisfy condition (6). The only edges that have fixed values are loop edges that do not belong to set A. □

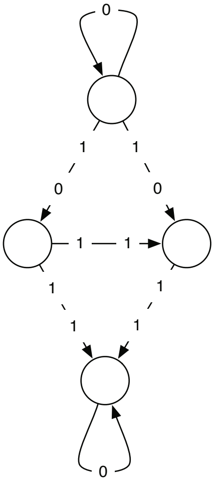


Fig. 4. A run

Recall that we use the notation $Inc(x)$ to denote the set of objects incident with either an edge x or a vertex x.

Lemma 5. $\sum_{e \in A} \sum_{v \in Inc(e)} r(e,v) \equiv 0 \pmod 2$, *for any run r of the parity protocol $\mathcal{P}_A$.*

Proof. Let $G = \langle V, E \rangle$. Using equations (6) and (5),

$$\sum_{e \in A} \sum_{v \in Inc(e)} r(e,v) \equiv \sum_{e \in E} \sum_{v \in Inc(e)} r(e,v) - \sum_{e \in E \setminus A} \sum_{v \in Inc(e)} r(e,v) \equiv$$
$$\equiv \sum_{v \in V} \sum_{e \in Inc(v)} r(e,v) - \sum_{e \notin A} 0 \equiv \sum_{v \in V} 0 - 0 \equiv 0 \pmod 2.$$

Everywhere below, by a path we will mean a sequence of edges that form a simple (undirected) path.

Definition 9. *For any path $\pi = e_0, e_1, \ldots, e_n$ in a graph G and any run r of the parity protocol $\mathcal{P}_A$, we define run r_π as*

$$r_\pi(e,v) = \begin{cases} 1 - r(e,v) & \text{if } v \in Inc(e_i) \cap Inc(e_{i+1}) \text{ for some } i < n, \\ r(e,v) & \text{otherwise.} \end{cases}$$

Informally, r_π is obtained from r by “flipping” the boolean values on path π at π's “internal” vertices. If a path is cyclic, then all vertices along this path are considered to be internal.

Lemma 6. *For any $r \in \mathcal{P}_A$ and any path π, if π is a cycle or starts and ends with edges that belong to set A, then $r_\pi \in \mathcal{R}(\mathcal{P}_A)$.*

Proof. Run r_π satisfies condition (5) because r_π is different from r at both ends of any non-terminal edge of path π. The same run r_π satisfies condition (6) at every vertex v of the graph, because path π includes either zero or two ends of edges incident at vertex v. □

Lemma 7. *If $|A| > 1$ and graph G is connected, then for any $e \in A$ and any $g \in \{0, 1\}$ there is a run $r \in \mathcal{R}(\mathcal{P}_A)$ such that $\sum_{v \in Inc(e)} r(e, v) \equiv g \pmod 2$.*

Proof. Let $\hat{r}(e, v)$ be a run of the protocol $\mathcal{P}_A$ which is equal to 0 for each end v of each edge e. If $g = 0$, then $\hat{r}$ is the required run r. Assume now that $g = 1$. Since $|A| > 1$ and graph G is connected, there is a path π that connects edge e with an edge $a \in A$ such that $a \neq e$. Notice that $\hat{r}_\pi$ is the desired run r, since $\sum_{v \in Inc(e)} \hat{r}_\pi(e, v) = \sum_{v \in Inc(e)} \hat{r}(e, v) + 1 \equiv g \pmod 2$. □

Lemma 8. *If $|A| > 1$ and graph G is connected, then $\mathcal{P}_A \nvDash [A]$.*

Proof. Let $A = \{a_1, \dots, a_k\}$. Pick any boolean values $g_1, \dots, g_k$ such that $g_1 + \dots + g_k \equiv 1 \pmod 2$. By Lemma 7, there are runs $r_1, \dots, r_k \in \mathcal{R}(\mathcal{P}_A)$ such that $\sum_{v \in a_i} r_i(a_i, v) \equiv g_i \pmod 2$ for any $i \leq k$. If $\mathcal{P}_A \vDash [A]$, then there is a run $r \in \mathcal{R}(\mathcal{P}_A)$ such that $r(a_i, v) = r_i(a_i, v)$ for each $v \in a_i$ and each $i \leq k$. Therefore, $\sum_{v \in a_1} r(a_1, v) + \dots + \sum_{v \in a_k} r(a_k, v) = \sum_{v \in a_1} r_1(a_1, v) + \dots + \sum_{v \in a_k} r_k(a_k, v) \equiv g_1 + \dots + g_k \equiv 1 \pmod 2$. This contradicts Lemma 5. □

Lemma 9. *If A and B are sets of edges of a graph $G = \langle V, E \rangle$, such that each connected component of the graph $\langle V, E \setminus B \rangle$ contains at least one edge from A, then $\mathcal{P}_A \vDash [B]$.*

Proofs of Lemma 9 and Lemma 10 from the next section are given in an extended version of this paper available on the third author's web page.[1]

7.2 Recursive Construction

In this section we will generalize the parity protocol through a recursive construction. First, however, we will establish a technical result that we will need for this construction.

Lemma 10 (protocol extension). *For any cut (X, Y) of graph $G = \langle V, E \rangle$ and any finite protocol $\mathcal{P}'$ on truncation G_X, there is a finite protocol $\mathcal{P}$ on G such that for any set $Q \subseteq E$, $\mathcal{P} \vDash [Q]$ if and only if $\mathcal{P}' \vDash [Q \cap E(G_X)]$.*

[1] http://www2.mcdaniel.edu/pnaumov

Lemma 11. *For any sets $A, B_1, \ldots, B_n$ of edges of G, if $G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A]$, then there is a finite protocol $\mathcal{P}$ over G such that $\mathcal{P} \vDash [B_i]$ for all $1 \le i \le n$ and $\mathcal{P} \nvDash [A]$.*

Proof. We use induction on the number of vertices of graph G.

Case 1. If $|A| \le 1$, then, by the Small Set axiom, $G \vdash [A]$. Hence, $G \vdash \bigwedge_{1\le i\le n}[B_i] \to [A]$, which is a contradiction.

Case 2. Suppose that the edges of graph G can be partitioned into two non-trivial disconnected sets X and Y. That is, no edge in X is adjacent with a edge in Y. Thus, the empty set is a gateway between $A \cap X$ and $A \cap Y$. By the Gateway axiom, $G \vdash [A \cap X] \to ([A \cap Y] \to [A])$. Hence, taking into account the assumption $G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A]$, either $G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A \cap X]$ or $G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A \cap Y]$. Without loss of generality, we will assume the former. By Theorem 1, $G \nvdash \bigwedge_{1\le i\le n}[B_i \cap X] \to [A \cap X]$. Consider the sets P_X and P_Y of all vertices in components X and Y respectively. Note that (P_X, P_Y) is a cut of G that has no crossing edges. Let G_X be the result of the truncation of G along this cut. By the Directed Truncation rule, $G_X \nvdash \bigwedge_{1\le i\le n}[B_i \cap X] \to [A \cap X]$. By the Induction Hypothesis, there is a protocol $\mathcal{P}'$ on G_X such that $\mathcal{P}' \nvDash [A \cap X]$ and $\mathcal{P}' \vDash [B_i \cap X]$, for any $i \le n$. Therefore, by Lemma 10, there is a protocol $\mathcal{P}$ on G such that $\mathcal{P} \nvDash [A]$ and $\mathcal{P} \vDash [B_i]$ for any $i \le n$.

Case 3. Suppose that graph G has a non-trivial directed cut (X, Y) such that $E(Y) \cap A = \varnothing$. Thus, by Theorem 1, $G \nvdash \bigwedge_{1\le i\le n}[B_i \cap E(X)] \to [A]$. By the Directed Truncation rule, $G_X \nvdash \bigwedge_{1\le i\le n}[B_i \cap E(X)] \to [A]$. By the Induction Hypothesis, there is a protocol $\mathcal{P}'$ over G_X such that $\mathcal{P}' \vDash [B_i \cap E(X)]$ for all $1 \le i \le n$ and $\mathcal{P}' \nvDash [A]$. Therefore, by Lemma 10, there is a protocol $\mathcal{P}$ on G such that $\mathcal{P} \nvDash [A]$ and $\mathcal{P} \vDash [B_i]$ for any $i \le n$.

Case 4. Suppose there is $i_0 \le n$ such that if all edges in B_{i_0} are removed from graph G, then at least one connected component of the resulting network G' does not contain an element of A. We will denote this connected component by Q. Let $W \subseteq B_{i_0}$ be the set of edges in G that connect a vertex from Q with a vertex not in Q. Any path connecting a edge in $E(Q)$ with a edge not in $E(Q)$ will have to contain a edge from W. In other words, W is a gateway between $E(Q)$ and the complement of $E(Q)$ in G. Hence, W is also a gateway between $A \cap E(Q)$ and $A \setminus E(Q)$. Therefore, by the Gateway axiom, taking into account that $(A \cap E(Q)) \cap W \subseteq E(Q) \cap W = \varnothing$,

$$G \vdash [A \cap E(Q), W] \to ([A \setminus E(Q)] \to [A]). \tag{7}$$

Recall now that by the assumption of this case, component Q of graph G' does not contain any elements of A. Hence, $A \cap E(Q) \subseteq B_{i_0}$. At the same time, $W \subseteq B_{i_0}$. Thus, from statement (7) and Theorem 1,

$$G \vdash [B_{i_0}] \to ([A \setminus E(Q))] \to [A]). \tag{8}$$

By the assumption of the lemma,

$$G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A]. \tag{9}$$

From statements (8) and (9), $G \nvdash \bigwedge_{1\le i\le n}[B_i] \to [A \setminus E(Q))]$. By the laws of propositional logic, $G \nvdash [B_{i_0}] \to (\bigwedge_{1\le i\le n}[B_i] \to [A \setminus E(Q)])$. Note that if $\overline{Q}$ is the complement of set Q, then $(\overline{Q}, Q)$ is a cut of graph G and W is the set of all crossing edges of this cut. Since $W \subseteq B_{i_0}$, by Theorem 1, $G \nvdash [W] \to (\bigwedge_{1\le i\le n}[B_i] \to [A \setminus E(Q)])$. Again by Theorem 1, $G \nvdash [W] \to (\bigwedge_{1\le i\le n}[B_i \setminus E(Q)] \to [A \setminus E(Q)])$. Let $G_{\overline{Q}}$ be the truncation of graph G along the cut $(\overline{Q}, Q)$. By the Truncation rule, $G_{\overline{Q}} \nvdash \bigwedge_{1\le i\le n}[B_i \setminus E(Q)] \to [A \setminus E(Q)]$.

By the Induction Hypothesis, there is a protocol $\mathcal{P}'$ on $G_{\overline{Q}}$ such that $\mathcal{P}' \nvDash [A \setminus E(Q)]$ and $\mathcal{P}' \vDash [B_i \setminus E(Q)]$ for any $i \le n$. Therefore, by Lemma 10, there is a protocol $\mathcal{P}$ on G such that $\mathcal{P} \nvDash [A]$ and $\mathcal{P} \vDash [B_i]$ for any $i \le n$.

Case 5. Assume now that (i) $|A| > 1$, (ii) graph G is connected, (iii) graph G has no non-trivial directed cuts (X, Y) such that $E(Y) \cap A = \varnothing$, and (iv) for any $i \le n$, if graph G' is obtained from G by the removal of all edges in B_i then each connected component of G' contains at least one element of A. Note that condition (iii) implies that A contains at least one loop edge at every sink vertex in graph G. Consider the parity protocol $\mathcal{P}_A$ over G. By Lemma 8, $\mathcal{P}_A \nvDash [A]$. By Lemma 9, $\mathcal{P}_A \vDash [B_i]$ for any $i \le n$. □

7.3 Protocol Composition

In this section, we define a composition of several protocols and finish the proof of the completeness theorem.

Definition 10. *For any protocols $\mathcal{P}^1 = (M^1, \Delta^1), \dots, \mathcal{P}^n = (M^n, \Delta^n)$ over a graph G, we define the Cartesian composition $\mathcal{P}^1 \times \mathcal{P}^2 \times \dots \times \mathcal{P}^n$ to be a pair (M, Δ) such that*

1. $M(e) = M^1(e) \times \dots \times M^n(e)$,
2. $\Delta_p(\langle e_1^1, \dots, e_1^n\rangle, \dots, \langle e_k^1, \dots, e_k^n\rangle) = \bigwedge_{1\le i\le n} \Delta_p^i(e_1^i, \dots, e_k^i)$.

For each composition $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \times \dots \times \mathcal{P}^n$, let $\{r(e)\}_i$ denote the ith component of the value of secret e over run r.

Lemma 12. *For any $n > 0$ and any finite protocols $\mathcal{P}^1, \dots, \mathcal{P}^n$ over a graph G, $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \times \dots \times \mathcal{P}^n$ is a finite protocol over G.*

Proof. The validity of the continuity condition for $\mathcal{P}$ follows from the continuity conditions for protocols $\mathcal{P}^1, \dots, \mathcal{P}^n$. □

Lemma 13. *For any $n > 0$, for any protocol $\mathcal{P} = \mathcal{P}^1 \times \mathcal{P}^2 \times \dots \times \mathcal{P}^n$ over a graph $G = \langle V, E\rangle$, and for any set of edges Q, $\mathcal{P} \vDash [Q]$ if and only if $\forall i\ (\mathcal{P}^i \vDash [Q])$.*

Proof. Let $Q = \{q_1, \dots, q_\ell\}$.
$(\Rightarrow)$: Assume $\mathcal{P} \vDash [Q]$ and pick any $i_0 \in \{1, \dots, n\}$. We will show that $\mathcal{P}^{i_0} \vDash [Q]$. Pick any runs $r_1', \dots, r_\ell' \in \mathcal{R}(\mathcal{P}^{i_0})$. For each $i \in \{1, \dots, i_0 - 1, i_0 + 1, \dots, n\}$, select an arbitrary run $r^i \in \mathcal{R}(\mathcal{P}^i)$. Such runs exist because graph G is acyclic and all

protocols satisfy the continuity condition. We then define a series of composed runs r_j for $j \in \{1, \dots, \ell\}$ by

$$r_j(e) = \langle r^1(e), \dots, r^{i_0-1}(e), r'_j(e), r^{i_0+1}(e), \dots, r^n(e)\rangle,$$

for each edge $e \in E$. Since the component parts of each r_j belong in their respective sets $\mathcal{R}(\mathcal{P}^i)$, the composed runs are themselves members of $\mathcal{R}(\mathcal{P})$. By our assumption, $\mathcal{P} \vDash [Q]$, thus there is $r \in \mathcal{R}(\mathcal{P})$ such that $r(q_i) = r_i(q_i)$ for any $i_0 \in \{1, \dots, \ell\}$. Finally, we consider the run r^*, where $r^*(e) = \{r(e)\}_{i_0}$ for each $e \in E$. That is, we let the value of r^* on e be the i_o^{th} component of $r(e)$. By the definition of composition, $r^* \in \mathcal{R}(\mathcal{P}^{i_0})$, and it matches the original $r'_1, \dots, r'_\ell \in \mathcal{R}(\mathcal{P}^{i_0})$ on edges $q_1, \dots, q_\ell$, respectively. Hence, we have shown that $\mathcal{P}^{i_0} \vDash [Q]$.
$(\Leftarrow)$: Assume $\forall i\ (\mathcal{P}^i \vDash [Q])$. We will show that $\mathcal{P} \vDash [Q]$. Pick any runs $r_1, \dots, r_\ell \in \mathcal{R}(\mathcal{P})$. For each $i \in \{1, \dots, n\}$, each $j \in \{1, \dots, \ell\}$, and each edge e, let $r^i_j(e) = \{r_j(e)\}_i$. That is, for each e, define a run r^i_j whose value on edge e equals the ith component of $r_j(e)$. Note that by the definition of composition, for each i and each j, r^i_j is a run in $\mathcal{R}(\mathcal{P}^i)$. Next, for each $i \in \{1, \dots, n\}$, we use the fact that $\mathcal{P}^i \vDash [Q]$ to construct a run $r^i \in \mathcal{R}(\mathcal{P}^i)$ such that $r^i(q_j) = r^i_j(q_j)$. Finally, we compose these n runs $r^1, \dots, r^n$ to get run $r \in \mathcal{R}(\mathcal{P})$. We note that the value of each edge q_j on r matches the the value of q_j in run $r_j \in \mathcal{R}(\mathcal{P})$, demonstrating that $\mathcal{P} \vDash [Q]$. □

We are now ready to prove the completeness theorem, which was stated earlier as Theorem 6:

Theorem 6. *For any graph $G = \langle V, E\rangle$, if $\mathcal{P} \vDash \phi$ for all finite protocols $\mathcal{P}$ over G, then $G \vdash \phi$.*

Proof. We give a proof by contradiction. Let X be a maximal consistent set of formulas from $\Phi(G)$ that contains $\neg\phi$. Let $\{A_1, \dots, A_n\} = \{A \subseteq E \mid [A] \notin X\}$ and $\{B_1, \dots, B_k\} = \{B \subseteq E \mid [B] \in X\}$. Thus, due to the maximality of set X, we have $G \nvdash \bigwedge_{1 \le j \le k}[B_j] \to [A_i]$, for every $i \in \{1, \dots, n\}$. We will construct a protocol $\mathcal{P}$ such that $\mathcal{P} \nvDash [A_i]$ for any $i \in \{1, \dots, n\}$ and $\mathcal{P} \vDash [B_j]$ for any $j \in \{1, \dots, k\}$.

First consider the case where $n = 0$. Pick any symbol ϵ and define $\mathcal{P}$ to be $\langle M, \Delta\rangle$ such that $M(e) = \{\epsilon\}$ for any $e \in E$ and action relation Δ_p to be the constant *True* at any vertex p. By Definition 4, $\mathcal{P} \vDash [C]$ for any $C \subseteq E$.

We will assume now that $n > 0$. By Theorem 11, there are finite protocols $\mathcal{P}^1, \dots, \mathcal{P}^n$ such that $\mathcal{P}^i \nvDash [A_i]$ and $\mathcal{P}^i \vDash [B_j]$ for all $j \in \{1, \dots, k\}$. Consider the composition $\mathcal{P}$ of protocols $\mathcal{P}^1, \dots, \mathcal{P}^n$. By Theorem 13, $\mathcal{P} \nvDash [A_i]$ for any $i \in \{1, \dots, n\}$ and $\mathcal{P} \vDash [B_j]$ for any $j \in \{1, \dots, j\}$.

Since X is a maximal consistent set, by induction on the structural complexity of any formula $\psi \in \Phi(G)$, one can show now that $\psi \in X$ if and only if $\mathcal{P} \vDash \psi$. Thus, $\mathcal{P} \vDash \neg\phi$. Therefore, $\mathcal{P} \nvDash \phi$, which is a contradiction. □

Corollary 1. *The set $\{(G, \phi) \mid G \vdash \phi\}$ is decidable.*

Proof. The complement of this set is recursively enumerable due to the completeness of the system with respect to finite protocols. □

8 Conclusion

In this paper, we captured the properties of information flow that can be described in terms of the independence relation $[A]$. This is not the only relation that can be used to describe properties of information flow on a graph. Another natural relation is the functional dependency relation $A \rhd B$ between two sets of edges. This relation is true if the values of edges in set A functionally determine the values of all edges in set B. A complete axiomatization of this relation when graph G is not fixed was given by Armstrong [5]. This logical system has become known in the database literature as Armstrong's axioms [6, p. 81]. Beeri, Fagin, and Howard [7] suggested a variation of Armstrong's axioms that describe properties of multi-valued dependency.

A complete axiomatization of relation $A \rhd B$ for a fixed *undirected* graph was given by More and Naumov [8]. It consists of Armstrong's axioms and a version of the Gateway axiom discussed in this paper, but contains no inference rules other than Modus Ponens. It appears, however, that this result can not be easily generalized to directed acyclic graphs. Thus, an axiomatization of relation $A \rhd B$ for directed acyclic graphs remains an open problem.

References

1. Sutherland, D.: A model of information. In: Proceedings of Ninth National Computer Security Conference, pp. 175–183 (1986)
2. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. 12(1), 1–47 (2008)
3. Miner More, S., Naumov, P.: On interdependence of secrets in collaboration networks. In: Proceedings of 12th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 208–217. Stanford University, Stanford (2009)
4. Miner More, S., Naumov, P.: Hypergraphs of multiparty secrets. In: Dix, J., Leite, J., Governatori, G., Jamroga, W. (eds.) CLIMA XI. LNCS (LNAI), vol. 6245, pp. 15–32. Springer, Heidelberg (2010)
5. Armstrong, W.W.: Dependency structures of data base relationships. In: Information Processing 1974, Proc. IFIP Congress, Stockholm, pp. 580–583. North-Holland, Amsterdam (1974)
6. Garcia-Molina, H., Ullman, J., Widom, J.: Database Systems: The Complete Book, 2nd edn. Prentice-Hall, Englewood Cliffs (2009)
7. Beeri, C., Fagin, R., Howard, J.H.: A complete axiomatization for functional and multivalued dependencies in database relations. In: SIGMOD 1977: Proceedings of the 1977 ACM SIGMOD International Conference on Management of Data, pp. 47–61. ACM, New York (1977)
8. Miner More, S., Naumov, P.: Functional dependence of secrets in a collaboration network. CoRR arXiv:1011.0399v1 [cs.LO] (2010)

The Boyce-Codd-Heath Normal Form for SQL

Flavio Ferrarotti[1], Sven Hartmann[2], Henning Köhler[3],
Sebastian Link[1], and Millist Vincent[4]

[1] School of Information Management, Victoria University of Wellington, New Zealand
[2] Institut für Informatik, Technische Universität Clausthal, Germany
[3] School of Information Technology & Electrical Engineering,
University of Queensland, Australia
[4] School of Computer and Information Science, University of South Australia,
Australia

Abstract. In the relational model of data the Boyce-Codd-Heath normal form, commonly just known as Boyce-Codd normal form, guarantees the elimination of data redundancy in terms of functional dependencies. For efficient means of data processing the industry standard SQL permits partial data and duplicate rows of data to occur in database systems. Consequently, the combined class of uniqueness constraints and functional dependencies is more expressive than the class of functional dependencies itself. Hence, the Boyce-Codd-Heath normal form is not suitable for SQL databases. We characterize the associated implication problem of the combined class in the presence of NOT NULL constraints axiomatically, algorithmically and logically. Based on these results we are able to establish a suitable normal form for SQL.

1 Introduction

In the *relational model of data* [7] a relation schema R denotes a finite set of attributes A that have a countably infinite domain $dom(A)$. A relation over R is a finite set of tuples, i.e. elements of the cartesian product over the domains. In addition, constraints restrict relations to those considered meaningful for the application. A functional dependency (FD) over R is an expression $X \to Y$ with $X, Y \subseteq R$. It restricts relations to those where every pair of tuples with the same values on all the attributes in X also has the same values on all the attributes in Y. FDs are essential for database design and data processing: if there is an FD $X \to Y$ over R with $Y \not\subseteq X$, then either all the attributes of $R - XY$ are also functionally dependent on X or there are relations with redundant data value occurrences. Redundancy can lead to inefficiencies with updates. A relation schema R is in *Boyce-Codd-Heath normal form* (BCHNF) [8,14,19] with respect to a given set Σ of FDs if for every FD $X \to Y$ in Σ, $Y \subseteq X$ or the FD $X \to R$ is implied by Σ. Our terminology pays tribute to Heath's contribution to the definition of what is commonly known as Boyce-Codd normal form in the literature. The definition was presented by Heath in a workshop organized by Codd in 1971 [14], and Heath himself acknowledges Codd's contribution to his work.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 110–122, 2011.

Example 1. Consider the relation schema Schedule with attributes *Location*, *Time*, and *Speaker*, and FD set Σ consisting of *Location, Time* → *Speaker*, and *Speaker, Time* → *Location*. Then Schedule is in BCHNF with respect to Σ. The following relation r on the left is an *Armstrong relation* for Σ. That is, r satisfies all the FDs in Σ and violates all the FDs not implied by Σ.

relation r

Location	*Time*	*Speaker*
Green Room	10am	Hilbert
Blue Room	10am	Gauss
Red Room	11am	Gauss
Red Room	01pm	Grothendieck
Red Room	02pm	Grothendieck

table t

Location	*Time*	*Speaker*
Green Room	10am	`ni`
Blue Room	10am	Gauss
Red Room	11am	Gauss
Red Room	01pm	Grothendieck
Red Room	02pm	Grothendieck
Red Room	02pm	Grothendieck

No data value occurrence in r is *redundant*: if we conceal any single value, then the remaining values and the FDs do not determine the concealed value. □

Commercial database systems deviate from the relational model of data. In the data definition and query standard *SQL* [9] database instances are tables where the column headers of the table correspond to attributes. The rows of the table correspond to tuples, but a table can contain different rows that have the same value in every column. Hence, an SQL table is a *bag* of rows. This feature lowers the cost of data processing as duplicate elimination is considered expensive. Furthermore, a so-called *null value*, marked `ni`, can occur in any column of any row in an SQL table. The null value indicates either non-existing, or existing but unknown, information. This feature of SQL makes it easy to enter new information into the database, since information is not always complete in practice. Null value occurrences can be forbidden for entire columns by declaring the corresponding column header `NOT NULL`. With these new features in mind we now revisit Example 1.

Example 2. Consider the SQL table Schedule from Example 1 with the same set Σ of constraints and where the column headers *Time* and *Location* are `NOT NULL`. The SQL table t from Example 1 on the right is an Armstrong table for Σ and the `NOT NULL` constraints. The BCHNF condition does not guarantee the absence of redundant data value occurrences over SQL tables. For example, the value of *Grothendieck* in the last row of table t is redundant: it is determined by the remaining values in the table t and the FD *Location, Time* → *Speaker*. □

Another important class of constraints over tables are *uniqueness constraints* (UCs). The UC $unique(X)$ restricts tables to those that do not have two distinct rows that are non-null and equal on every attribute in X. In the relational model UCs are not studied separately because any set of tuples over R satisfies the UC $unique(X)$ if and only if it satisfies the FD $X \rightarrow R$. However, this equivalence no longer holds over SQL tables, as illustrated in Example 2. Indeed, if $X = \{Location, Time\}$, then table t satisfies $X \rightarrow$ Schedule, but not $unique(X)$. This means that, in the context of SQL tables, the combined class of UCs and FDs should be studied, preferably in the context of `NOT NULL` constraints. Moreover, Example 2 motivates our pursuit of a normal form condition for SQL table definitions that eliminates redundant data value occurrences.

Contributions and Organization. We summarize previous work in Section 2 and give preliminary definitions in Section 3. In Section 4 we establish a finite axiomatization for the combined class of UCs and FDs in the presence of `NOT NULL` constraints. In Section 5 we show that the implication problem of this class is equivalent to that of goal and definite clauses in Cadoli and Schaerf's para-consistent family of $\mathcal{S}$-3 logics. In Section 6 we propose a new syntactic normal form condition for SQL table definitions. Finally, in Section 7 we justify our condition semantically by showing that it is necessary and sufficient for the absence of redundant data value occurrences in any SQL tables. We also show that our condition can be checked in time quadratic in the input, and is independent of the representation of the constraints. We conclude in Section 8.

2 Related Work

Data dependencies and normalization are essential to the design of the target database, the maintenance of the database during its lifetime, and all major data processing tasks, cf. [1].

In the relational model, a UC $unique(X)$ over relation schema R is satisfied by a relation if and only if the relation satisfies the FD $X \rightarrow R$. Hence, in this context it suffices to study the class of FDs alone. Armstrong [4] established the first axiomatization for FDs. The implication problem of FDs can be decided in time linear in the input [10]. Boyce and Codd [8] and Heath [14] introduced what is now known as the Boyce-Codd-Heath normal form for relation schemata. Vincent showed that BCHNF is a sufficient and necessary condition to eliminate all possible redundant data value occurrences as well as data processing difficulties in terms of FDs [22]. Arenas and Libkin also justified the BCHNF condition in terms of information-theoretic measures [3].

One of the most important extensions of Codd's basic relational model [7] is incomplete information [15]. This is mainly due to the high demand for the correct handling of such information in real-world applications. While there are several possible interpretations of a null value, most of the previous work on data dependencies is based on Zaniolo's no information interpretation [24]. Atzeni and Morfuni established an axiomatization of FDs in the presence of `NOT NULL` constraints under the no information interpretation [5]. They did not consider bags, which commonly appear in SQL, nor normalization. Köhler and Link investigated UCs and FDs over bags, but did not consider null values [17]. Their results are subsumed by the current contributions as the special case where every column header is declared `NOT NULL`. Finally, Hartmann and Link established the equivalence of the implication problem for the combined class of FDs and multivalued dependencies in the presence of `NOT NULL` constraints to that of a propositional fragment of Cadoli and Schaerf's family of $\mathcal{S}$-3 logics [13]. However, they only looked at relations, where UCs are already subsumed by FDs, and did not consider bags. The paper [13] did also not consider any normal forms. The equivalences from [13] cover those by Sagiv et al. [20] established for the special case where $\mathcal{S}$ covers all variables.

3 SQL Table Definitions

We summarize the basic notions. Let $\mathfrak{A} = \{H_1, H_2, \ldots\}$ be a (countably) infinite set of distinct symbols, called (column) headers. An *SQL table definition* is a finite non-empty subset T of $\mathfrak{A}$. Each header H of a table definition T is associated with a countably infinite domain $dom(H)$ which represents the possible values that can occur in the column H denotes. To encompass incomplete information every column may have a null value, denoted by $\mathtt{ni} \in dom(H)$. The intention of $\mathtt{ni}$ is to mean "no information". This interpretation can therefore model non-existing as well as existing but unknown information [5,24].

For header sets X and Y we may write XY for $X \cup Y$. If $X = \{H_1, \ldots, H_m\}$, then we may write $H_1 \cdots H_m$ for X. In particular, we may write simply H to represent the singleton $\{H\}$. A *row* over T (T-row or simply row, if T is understood) is a function $r : T \to \bigcup_{H \in T} dom(H)$ with $r(H) \in dom(H)$ for all $H \in R$. The null value occurrence $r(H) = \mathtt{ni}$ associated with a header H in a row r means that no information is available about the header H for the row r. For $X \subseteq T$ let $r[H]$ denote the restriction of the row r over T to X. An *SQL table* t over T is a finite multi-set of rows over R. In particular, a table t over T may contain two rows r_1 and r_2 such that $r_1 \neq r_2$ and $r_1(H) = r_2(H)$ for all $H \in T$. For a row r over T and a set $X \subseteq T$, r is said to be X-total if for all $H \in X$, $r(H) \neq \mathtt{ni}$. Similar, a table t over T is said to be X-total, if every row r of t is X-total. A table t over T is said to be a *total table* if it is T-total.

Following the SQL standard a *uniqueness constraint* (UC) over an SQL table definition T is an expression *unique*(X) where $X \subseteq T$. An SQL table t over T is said to satisfy the uniqueness constraint *unique*(X) over T ($\models_t$ *unique*(X)) if and only if for all distinct rows $r_1, r_2 \in t$ the following holds: if r_1 and r_2 are X-total, then there is some $H \in X$ such that $r_1(H) \neq r_2(H)$.

Functional dependencies are important for the relational [7] and other data models [2,11,12,23]. Following Lien [18], a *functional dependency* (FD) over T is a statement $X \to Y$ where $X, Y \subseteq T$. The FD $X \to Y$ over T is satisfied by a table t over T ($\models_t X \to Y$) if and only if for all $r_1, r_2 \in t$ the following holds: if r_1 and r_2 are X-total and $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$. We call $X \to Y$ *trivial* whenever $Y \subseteq X$, and non-trivial otherwise. For total tables the FD definition reduces to the standard definition of a functional dependency [1], and so is a sound generalization. It is also consistent with the no-information interpretation [5,18].

Following Atzeni and Morfuni [5], a *null-free sub-definition* (NFS) over the table definition T is an expression T_s where $T_s \subseteq T$. The NFS T_s over T is satisfied by a table t over T ($\models_t T_s$) if and only if t is T_s-total. SQL allows the specification of column headers as `NOT NULL`. Hence, the set of headers declared `NOT NULL` forms the single NFS over the underlying SQL table definition.

For a set Σ of constraints over some table definition T, we say that a table t over T *satisfies* Σ ($\models_t \Sigma$) if t satisfies every $\sigma \in \Sigma$. If for some $\sigma \in \Sigma$ the table t does not satisfy σ we say that t violates σ (and violates Σ) and write $\not\models_t \sigma$ ($\not\models_t \Sigma$). We are interested in the combined class $\mathcal{C}$ of uniqueness constraints and FDs in the presence of an NFS.

Constraints interact with one another. Let T be an SQL table definition, let $T_s \subseteq T$ denote an NFS over T, and let $\Sigma \cup \{\varphi\}$ be a set of uniqueness constraints and FDs over T. We say that Σ *implies* φ in the presence of T_s ($\Sigma \models_{T_s} \varphi$) if every table t over T that satisfies Σ and T_s also satisfies φ. If Σ does not imply φ in the presence of T_s we may also write $\Sigma \not\models_{T_s} \varphi$. For Σ we let $\Sigma^*_{T_s} = \{\varphi \mid \Sigma \models_{T_s} \varphi\}$ be the *semantic closure* of Σ, i.e., the set of all uniqueness constraints and FDs implied by Σ in the presence of T_s. In order to determine the logical consequences we use a syntactic approach by applying inference rules, e.g. those in Table 1. These inference rules have the form

$$\frac{\text{premise}}{\text{conclusion}}\,\text{condition},$$

and inference rules without any premises are called axioms. An inference rule is called sound, if whenever the set of constraints in the premise of the rule and the NFS are satisfied by some table over T and the constraints and NFS satisfy the conditions of the rule, then the table also satisfies the constraint in the conclusion of the rule. We let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote the *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of constraints such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or results from an application of an inference rule in $\mathfrak{R}$ to some elements in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. For a finite set Σ, let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ be its *syntactic closure* under inferences by $\mathfrak{R}$. A set $\mathfrak{R}$ of inference rules is said to be *sound* (*complete*) for the implication of uniqueness constraints and FDs in the presence of an NFS if for every table definition T, for every NFS T_s over T and for every set Σ of uniqueness constraints and FDs over T we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{T_s}$ ($\Sigma^*_{T_s} \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* for the implication of uniqueness constraints and FDs in the presence of an NFS if $\mathfrak{R}$ is both sound and complete.

Example 3. The SQL table in Example 2 satisfies the FD *Location, Time* → *Speaker*, but violates the UC *unique*(*Location, Time*). The table

Location	*Time*	*Speaker*
Red Room	ni	Gauss
Red Room	ni	Grothendieck

satisfies the NFS {*Location, Speaker*}, the UC *unique*(*Location, Time*) and the FDs *Location* → *Time* and *Time* → *Speaker*. The table violates the NFS {*Time*}, the UC *unique*(*Location*) and the FD *Location* → *Speaker*. □

4 Axiomatic and Algorithmic Characterization

Let $\mathfrak{S}$ denote the set of inference rules in Table 1. The soundness of the rules in $\mathfrak{S}$ is not difficult to show. For the completeness of $\mathfrak{S}$ we use the result that

Table 1. Axiomatization of UCs and FDs in the presence of an NFS

$\dfrac{\mathit{unique}(X)}{X \rightarrow Y}$ (demotion)	$\dfrac{}{XY \rightarrow X}$ (reflexivity)	$\dfrac{X \rightarrow YZ}{X \rightarrow Y}$ (decomposition)
$\dfrac{X \rightarrow Y \quad \mathit{unique}(Y)}{\mathit{unique}(X)} Y \subseteq XT_s$ (null pullback)	$\dfrac{X \rightarrow Y \quad Y \rightarrow Z}{X \rightarrow Z} Y \subseteq XT_s$ (null transitivity)	$\dfrac{X \rightarrow Y \quad X \rightarrow Z}{X \rightarrow YZ}$ (union)

the set $\mathfrak{M}$ consisting of the reflexivity axiom, the union, decomposition and null transitivity rule is sound and complete for FDs in the presence of an NFS [5]. In fact, the completeness of $\mathfrak{S}$ follows from that of $\mathfrak{M}$ and the following lemma. For a set $\Sigma = \Sigma_{\text{UC}} \cup \Sigma_{\text{FD}}$ of UCs and FDs over table definition T let $\Sigma_{\text{UC}}^{\text{FD}} = \{X \rightarrow T \mid \mathit{unique}(X) \in \Sigma_{\text{UC}}\}$ be the set of FDs associated with Σ_{UC} and let $\Sigma[\text{FD}] := \Sigma_{\text{UC}}^{\text{FD}} \cup \Sigma_{\text{FD}}$ be the set of FDs associated with Σ.

Lemma 1. *Let T be an SQL table definition, T_s an NFS, and Σ a set of UCs and FDs over T. Then the following hold:*

1. *$\Sigma \models_{T_s} X \rightarrow Y$ if and only if $\Sigma[FD] \models_{T_s} X \rightarrow Y$,*
2. *$\Sigma \models_{T_s} \mathit{unique}(X)$ if and only if $\Sigma[FD] \models_{T_s} X \rightarrow T$ and there is some $\mathit{unique}(Z) \in \Sigma$ such that $Z \subseteq XT_s$.* □

Theorem 1. *The set $\mathfrak{S}$ is a finite axiomatization for the implication of UCs and FDs in the presence of an NFS.*

Proof (Sketch). We sketch the completeness of $\mathfrak{S}$ by showing that for an arbitrary table definition T, an arbitrary NFS T_s and an arbitrary set $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \cup \{\varphi\}$ of uniqueness constraints and functional dependencies over T the following holds: if $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \models_{T_s} \varphi$, then $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \vdash_{\mathfrak{S}} \varphi$. We consider two cases. In case (1) φ denotes the FD $X \rightarrow Y$. Then we know by Lemma 1 that $\Sigma[\text{FD}] \models_{T_s} \varphi$ holds. From the completeness of $\mathfrak{M}$ for the implication of functional dependencies in the presence of an NFS we conclude that $\Sigma[\text{FD}] \vdash_{\mathfrak{M}} \varphi$. Since $\mathfrak{M} \subseteq \mathfrak{S}$ holds we know that $\Sigma[\text{FD}] \vdash_{\mathfrak{S}} \varphi$ holds, too. The *demotion rule* shows for all $\sigma \in \Sigma_{\text{UC}}^{\text{FD}}$ that $\Sigma_{\text{UC}} \vdash_{\mathfrak{S}} \sigma$ holds. Consequently, we have $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \vdash_{\mathfrak{S}} \varphi$. This concludes case (1). In case (2) φ denotes the UC $\mathit{unique}(X)$. From $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \models_{T_s} \mathit{unique}(X)$ we conclude by Lemma 1 that there is some $\mathit{unique}(Z) \in \Sigma_{\text{UC}}$ such that $Z \subseteq XT_s$ holds. We also conclude from $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \models_{T_s} \mathit{unique}(X)$ that $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \models_{T_s} X \rightarrow Z$ holds by soundness of the demotion rule. From case (1) it follows that $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \vdash_{\mathfrak{S}} X \rightarrow Z$ holds. A final application of the *null pullback rule* shows that $\Sigma_{\text{UC}} \cup \Sigma_{\text{FD}} \vdash_{\mathfrak{S}} \varphi$ holds. □

Lemma 1 establishes an algorithmic characterization of the associated implication problem. In fact, it suffices to compute the *header set closure* $X^*_{\Sigma[\text{FD}],T_s} := \{H \in T \mid \Sigma[\text{FD}] \models_{T_s} X \rightarrow H\}$ of X with respect to $\Sigma[\text{FD}]$ and T_s [5]. The size

$|\varphi|$ of φ is the total number of attributes occurring in φ, and the size $||\Sigma||$ of Σ is the sum of $|\sigma|$ over all elements $\sigma \in \Sigma$.

Theorem 2. *The problem whether a UC or FD φ is implied by a set Σ of UCs and FDs can be decided in $\mathcal{O}(||\Sigma \cup \{\varphi\}||)$ time.* □

5 Equivalence to Goal and Definite Clauses in $\mathcal{S}$-3 Logics

Here we refine the correspondence between the implication of FDs in the presence of NFSs and the implication of Horn clauses in Cadoli and Schaerf's family of $\mathcal{S}$-3 logics, established for tables that are sets of rows [13].

$\mathcal{S}$-3 semantics. Schaerf and Cadoli [21] introduced $\mathcal{S}$-3 logics as "a semantically well-founded logical framework for sound approximate reasoning, which is justifiable from the intuitive point of view, and to provide fast algorithms for dealing with it even when using expressive languages".

For a finite set $\mathcal{L}$ of propositional variables let $\mathcal{L}^\ell$ denote the set of all literals over $\mathcal{L}$, i.e., $\mathcal{L}^\ell = \mathcal{L} \cup \{\neg H' \mid H' \in \mathcal{L}\} \subseteq \mathcal{L}^*$ where $\mathcal{L}^*$ denotes the propositional language over $\mathcal{L}$. Let $\mathcal{S} \subseteq \mathcal{L}$. An $\mathcal{S}$-3 interpretation of $\mathcal{L}$ is a total function $\hat{\omega} : \mathcal{L}^\ell \to \{\mathbb{F}, \mathbb{T}\}$ that maps every variable $H' \in \mathcal{S}$ and its negation $\neg H'$ into opposite values ($\hat{\omega}(H') = \mathbb{T}$ if and only if $\hat{\omega}(\neg H') = \mathbb{F}$), and that does not map both a variable $H' \in \mathcal{L} - \mathcal{S}$ and its negation $\neg H'$ into $\mathbb{F}$ (we must not have $\hat{\omega}(H') = \mathbb{F} = \hat{\omega}(\neg H')$ for any $H' \in \mathcal{L} - \mathcal{S}$). An $\mathcal{S}$-3 interpretation $\hat{\omega} : \mathcal{L}^\ell \to \{\mathbb{F}, \mathbb{T}\}$ of $\mathcal{L}$ can be lifted to a total function $\hat{\Omega} : \mathcal{L}^* \to \{\mathbb{F}, \mathbb{T}\}$ by means of simple rules [21]. Since we are only interested in Horn clauses here we require the following two rules for assigning truth values to a Horn clause: (1) $\hat{\Omega}(\varphi') = \hat{\omega}(\varphi')$, if $\varphi' \in \mathcal{L}^\ell$, and (2) $\hat{\Omega}(\varphi' \vee \psi') = \mathbb{T}$, if $\hat{\Omega}(\varphi') = \mathbb{T}$ or $\hat{\Omega}(\psi') = \mathbb{T}$. An $\mathcal{S}$-3 interpretation $\hat{\omega}$ is a *model* of a set Σ' of $\mathcal{L}$-formulae, if $\hat{\Omega}(\sigma') = \mathbb{T}$ holds for every $\sigma' \in \Sigma'$. We say that Σ' *$\mathcal{S}$-3 implies* an $\mathcal{L}$-formula φ', denoted by $\Sigma' \models^3_{\mathcal{S}} \varphi'$, if every $\mathcal{S}$-3 interpretation that is a model of Σ' is also a model of φ'.

Mappings between constraints and formulae. In the first step, we define the fragment of $\mathcal{L}$-formulae that corresponds to UCs and FDs in the presence of an NFS T_s over a table definition T. Let $\phi : T \to \mathcal{L}$ denote a bijection between T and the set $\mathcal{L} = \{H' \mid H \in T\}$ of propositional variables that corresponds to T. For an NFS T_s over T let $\mathcal{S} = \phi(T_s)$ be the set of propositional variables in $\mathcal{L}$ that corresponds to T_s. Hence, the variables in $\mathcal{S}$ are the images of those column headers of T declared `NOT NULL`. We now extend ϕ to a mapping Φ from the set of UCs and FDs over T. For a UC $\mathit{unique}(H_1, \ldots, H_n)$ over T, let $\Phi(\mathit{unique}(H_1, \ldots, H_n))$ denote the goal clause $\neg H'_1 \vee \cdots \vee \neg H'_n$. For an FD $H_1, \ldots, H_n \to H$ over T, let $\Phi(H_1, \ldots, H_n \to H)$ denote the definite clause $\neg H'_1 \vee \cdots \vee \neg H'_n \vee H'$. For the sake of presentation, but without loss of generality, we assume that FDs have only a single column header on their right-hand side. As usual, disjunctions over zero disjuncts are interpreted as $\mathbb{F}$. In what follows, we may simply denote $\Phi(\varphi) = \varphi'$ and $\Phi(\Sigma) = \{\sigma' \mid \sigma \in \Sigma\} = \Sigma'$.

The equivalence. Our aim is to show that for every SQL table definition T, for every set $\Sigma \cup \{\varphi\}$ of UCs and FDs and for every NFS T_s over T, there is some T_s-total table t that satisfies Σ and violates φ if and only if there is an $\mathcal{S}$-3 model

$\hat{\omega}_t$ of Σ' that is not an $\mathcal{S}$-3 model of φ'. For an arbitrary table t it is not obvious how to define the $\mathcal{S}$-3 interpretation $\hat{\omega}_t$. However, for deciding the implication problem $\Sigma \models_{T_s} \varphi$ it suffices to examine two-row tables, instead of arbitrary tables. For two-row tables $\{r_1, r_2\}$ we define the *special-3-interpretation* of $\mathcal{L}$ by

- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{F}$, if $\texttt{ni} \neq r_1(H) = r_2(H) \neq \texttt{ni}$,
- $\hat{\omega}_{\{r_1,r_2\}}(H') = \mathbb{T}$ and $\hat{\omega}_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $r_1(H) = \texttt{ni} = r_2(H)$,
- $\hat{\omega}'_{\{r_1,r_2\}}(H') = \mathbb{F}$ and $\hat{\omega}'_{\{r_1,r_2\}}(\neg H') = \mathbb{T}$, if $r_1(H) \neq r_2(H)$

for all $H' \in \mathcal{L}$. If $\{r_1, r_2\}$ is T_s-total, then $\hat{\omega}_{\{r_1,r_2\}}$ is an $\mathcal{S}$-3 interpretation.

Theorem 3. *Let $\Sigma \cup \{\varphi\}$ be a set of UCs and FDs over the SQL table definition T, and let T_s denote an NFS over T. Let $\mathcal{L}$ denote the set of propositional variables that corresponds to T, $\mathcal{S}$ the set of variables that corresponds to T_s, and $\Sigma' \cup \{\varphi'\}$ the set of goal and definite clauses over $\mathcal{L}$ that corresponds to $\Sigma \cup \{\varphi\}$. Then $\Sigma \models_{T_s} \varphi$ if and only if $\Sigma' \models^3_{\mathcal{S}} \varphi'$.* □

An example of the equivalence. Consider the table definition Schedule with $\text{Schedule}_s = \{\mathit{Location}\}$ and $\Sigma = \{\mathit{Speaker} \to \mathit{Location}, \mathit{Location} \to \mathit{Time}\}$. Suppose we wonder if the FD $\varphi_1 = \mathit{Speaker} \to \mathit{Time}$ is implied by Σ in the presence of Schedule_s. According to Theorem 3 the problem $\Sigma \models_{\text{Schedule}_s} \varphi_1$ is equivalent to $\Sigma' \models^3_{\mathcal{S}} \varphi'_1$ where $\mathcal{S} = \{\mathit{Location}'\}$. Suppose an $\mathcal{S}$-3 interpretation $\hat{\omega}$ is not a model of φ'_1. Then $\hat{\omega}(\neg \mathit{Speaker}') = \mathbb{F} = \hat{\omega}(\mathit{Time}')$. For $\hat{\omega}$ to be an $\mathcal{S}$-3 model of Σ' we must thus have $\hat{\omega}(\mathit{Location}') = \mathbb{T} = \hat{\omega}(\neg \mathit{Location}')$, but $\mathit{Location}' \in \mathcal{S}$. We conclude that $\Sigma' \models^3_{\mathcal{S}} \varphi'_1$ and by Theorem 3 also $\Sigma \models_{\text{Schedule}_s} \varphi_1$. Let now be $\varphi_2 = \mathit{unique}(\mathit{Speaker})$. Then $\Sigma \not\models_{T_s} \varphi_1$ as the following SQL table t demonstrates:

Speaker	*Location*	*Time*
Grothendieck	Red Room	ni
Grothendieck	Red Room	ni

.

Indeed, the special $\mathcal{S}$-3 interpretation $\hat{\omega}_t$ where for all $L \in \mathcal{L}^\ell$, $\hat{\omega}_t(L) = \mathbb{F}$ iff $L \in \{\neg \mathit{Speaker}', \neg \mathit{Location}'\}$ is a model of Σ' but not a model of φ'_2.

6 The Boyce-Codd-Heath Normal Form for SQL

Boyce and Codd [8] and Heath [14] introduced a normal form condition on relation schemata that characterizes the absence of certain processing difficulties with any relation over the schema [22]. For SQL table definitions, no normal forms have been proposed to the best of our knowledge. We now propose an extension of the classical Boyce-Codd-Heath normal form to SQL table definitions.

Definition 1. *Let T denote an SQL table definition, T_s a null-free subdefinition, and Σ a set of UCs and FDs over T. Then T is said to be in* Boyce-Codd-Heath normal form *with respect to Σ and T_s if and only if for all non-trivial functional dependencies $X \to Y \in \Sigma^+_{\mathfrak{S}}$ we have $\mathit{unique}(X) \in \Sigma^+_{\mathfrak{S}}$.* □

Schema SCHEDULE of Example 2 is not in BCHNF with respect to Σ and T_s. However, if we replace the two FDs in Σ by the two UCs *unique*(*Location, Time*) and *unique*(*Speaker*, *Time*), then SCHEDULE is indeed in BCHNF with respect to Σ and T_s. It is very important to note here that the UCs are much stronger than the FDs. If the FD $X \to H$ is meaningful over T, then the table definition with projected column header set $T' = XH$ still carries the FD $X \to H$, but the UC *unique*(X) may not be meaningful over T'. That is, decomposition and synthesis approaches [6,16,19] deserve new attention in the context of SQL. In general, duplicates should only be tolerated when they are meaningful, or updates are less expensive than duplicate elimination.

7 Semantic Justification

We will now justify our syntactic definition of BCHNF semantically by showing that the condition is sufficient and necessary for the absence of redundant data value occurrences in any future tables. Following Vincent [22] we will make the notion of data redundancy explicit. Let T be an SQL table definition, H a column header of T, and r a row over T. A *replacement* of $r(H)$ is a row r' over T that satisfies the following conditions: i) for all $H' \in T - \{H\}$ we have $r'(H') = r(H')$, and ii) $r'(H) \neq r(H)$. Intuitively, a data value occurrence in some Σ-satisfying table is redundant if the occurrence cannot be replaced by any other data value without violating some constraint in Σ.

Definition 2. *Let T be an SQL table definition, $H \in T$ a column header, T_s an NFS and Σ a set of UCs and FDs over T, t a table over T that satisfies Σ and T_s, and r a row in t. We say that the data value occurrence $r(H)$ is* redundant *if and only if* every *replacement r' of $r(H)$ results in a table $t' := (t - \{r\}) \cup \{r'\}$ that violates Σ. We say that T is in* Redundancy-Free Normal Form *(RFNF) with respect to Σ and T_s if and only if there is no table t over T such that i) t satisfies Σ and T_s, and ii) t contains a row r such that for some column header H of T the data value occurrence $r(H)$ is redundant.* □

We show that the syntactic BCHNF condition of Definition 1 captures the semantic RFNF condition of Definition 2.

Theorem 4. *Let T be an SQL table definition, T_s an NFS and Σ a set of UCs and FDs over T. Then T is in RFNF with respect to Σ and T_s if and only if T is in BCHNF with respect to Σ and T_s.*

Proof. Let T not be in RFNF with respect to Σ and T_s. Then there is some T_s-total table t over T that satisfies Σ, some row $r \in t$ and some header $H \in T$ such that $r(H)$ is redundant. We need to show that there is some non-trivial FD $X \to Y \in \Sigma^+_{\mathfrak{S}}$ such that *unique*$(X) \notin \Sigma^+_{\mathfrak{S}}$. Let $t[H] := \{\bar{r}(H) | \bar{r} \in t\}$. Define a replacement r' of $r(H)$ such that $r'(H) \in dom(H) - (t[H] \cup \{\texttt{ni}\})$. Furthermore, let $t' := (t - \{r\}) \cup \{r'\}$. Since $r(H)$ is redundant it follows that t' violates Σ. Since $\models_t \Sigma$ and t' agrees with t except on $r'(H) \notin t[H] \cup \{\texttt{ni}\}$ it follows that

t' cannot violate any UC in Σ. Let t' violate the FD $X \to Y \in \Sigma$. From the definition of r' and the properties of t and t' it follows that $H \in Y - X$. Hence, $X \to Y$ is non-trivial. Since t' violates $X \to Y \in \Sigma$ there is some $r'' \in t' - \{r'\}$ such that $r''[X] = r'[X]$, and r'', r' are X-total. Moreover, $r'' \in t$, $r''[X] = r[X]$ and r'', r are X-total since $H \notin X$. Therefore, t satisfies Σ but t violates the UC *unique*(X). Hence, *unique*$(X) \notin \Sigma^*_{T_s}$ and by the soundness of $\mathfrak{G}$ we conclude *unqiue*$(X) \notin \Sigma^+_{\mathfrak{G}}$. It follows that T is not in BCHNF with respect to Σ and T_s.

Vice versa, let T not be in BCHNF with respect to Σ and T_s. Then there is some non-trivial FD $X \to Y \in \Sigma^+_{\mathfrak{G}}$ such that *unique*$(X) \notin \Sigma^+_{\mathfrak{G}}$. We need to show that there is some table t over T that satisfies Σ and T_s, some row $r \in t$ and some header $H \in T$ such that $r(H)$ is redundant. Let $t := \{r, r'\}$ consist of two rows r and r' over T such that for all $H' \in T$, i) $\mathtt{ni} \neq r(H') = r'(H') \neq \mathtt{ni}$ holds if and only if $H' \in X(X^+_{\Sigma,T_s} \cap T_s)$, ii) $r(H') = \mathtt{ni} = r'(H')$ if and only if $H' \in X^+_{\Sigma,T_s} - XT_s$, and iii) $\mathtt{ni} \neq r(H') \neq r'(H') \neq \mathtt{ni}$ if and only if $H' \in T - X^+_{\Sigma,T_s}$. Here,

$$X^+_{\Sigma,T_s} := \{H' \in T | X \to H' \in \Sigma^+_{\mathfrak{G}}\}.$$

It follows immediately that t is T_s-total. We show that t satisfies Σ.

Let $U \to V \in \Sigma$ and let $r[U] = r'[U]$ such that r, r' are U-total. It follows that $U \subseteq X(X^+_{\Sigma,T_s} \cap T_s)$. From $X \to X^+_{\Sigma,T_s} \in \Sigma^+_{\mathfrak{G}}$ and $U \subseteq X^+_{\Sigma,T_s}$ follows $X \to U \in \Sigma^+_{\mathfrak{G}}$ by the *decomposition rule*. From $X \to U \in \Sigma^+_{\mathfrak{G}}$, $U \to V \in \Sigma$ and $U \subseteq XT_s$ we conclude $X \to V \in \Sigma^+_{\mathfrak{G}}$ by means of the *null transitivity rule*. Consequently, $V \subseteq X^+_{\Sigma,T_s}$ and therefore $r[V] = r'[V]$. We conclude that t satisfies $U \to V$.

Let *unique*$(U) \in \Sigma$, and assume that $r[U] = r'[U]$ holds for the distinct U-total rows r and r'. We conclude that $U \subseteq X(X^+_{\Sigma,T_s} \cap T_s)$ holds. From $X \to X^+_{\Sigma} \in \Sigma^+_{\mathfrak{G}}$ we infer $X \to U \in \Sigma^+_{\mathfrak{G}}$ by means of the *decomposition rule*. From *unique*$(U) \in \Sigma$, $X \to U \in \Sigma^+_{\mathfrak{G}}$ and $U \subseteq XT_s$ we infer that *unique*$(X) \in \Sigma^+_{\mathfrak{G}}$ by an application of the *null pullback rule*. This, however, is a contradiction since *unique*$(X) \notin \Sigma^+_{\mathfrak{G}}$. Consequently, t satisfies *unique*(U). Hence, t satisfies Σ.

Now let $H \in Y - X$. Since $Y \subseteq X^+_{\Sigma,T_s}$ it follows that $r(H)$ is redundant. Therefore, T is not in RFNF with respect to Σ and T_s. □

Definition 1 refers to the syntactic closure $\Sigma^+_{\mathfrak{G}}$ of Σ and T_s under $\mathfrak{G}$, which can be exponential in the size of Σ. Therefore, the question remains if the problem whether an SQL table definition is in BCHNF with respect to Σ and T_s can be decided efficiently.

Theorem 5. *Let T be an SQL table definition, T_s an NFS and Σ a set of UCs and FDs over T. Then the following conditions are equivalent:*

1. *T is in BCHNF with respect to Σ and T_s,*
2. *for all non-trivial FDs $X \to Y \in \Sigma$ we have: unique$(X) \in \Sigma^+_{\mathfrak{G}}$,*
3. *for all non-trivial FDs $X \to Y \in \Sigma$ we have: $X \to T \in \Sigma^+_{\mathfrak{G}}$ and there is some unique$(Z) \in \Sigma$ such that $Z \subseteq XT_s$.*

Proof. We show first the equivalence between *1.* and *2.* Condition *1.* implies condition *2.* since $\Sigma \subseteq \Sigma^+_{\mathfrak{S}}$. We show next that condition *2.* implies condition *1.* Assume that T is not in BCHNF with respect to Σ and T_s. That is, there is some non-trivial FD $X \to Y \in \Sigma^+_{\mathfrak{S}}$ such that $unique(X) \notin \Sigma^+_{\mathfrak{S}}$. We need to show that there is some non-trivial FD $X' \to Y' \in \Sigma$ such that $unique(X') \notin \Sigma^+_{\mathfrak{S}}$. Let

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma^+_{\mathfrak{S}}$$

be a proper chain where for all $j = 1, \ldots, k$ the set Σ_j results from Σ_{j-1} by a single application of an inference rule in $\mathfrak{S}$. We show that if there is some non-trivial FD $X \to Y \in \Sigma_j$ such that $unique(X) \notin \Sigma^+_{\mathfrak{S}}$, then there is some non-trivial FD $X' \to Y' \in \Sigma_{j-1}$ such that $unique(X') \notin \Sigma^+_{\mathfrak{S}}$. For $j > 0$ let $X \to Y \in \Sigma_j - \Sigma_{j-1}$ be non-trivial such that $unique(X) \notin \Sigma^+_{\mathfrak{S}}$. Then $X \to Y$ has been inferred either by means of the decomposition, union or null transitivity rule. In case of the decomposition rule we have $X \to YZ \in \Sigma_{j-1}$ with $unique(X) \notin \Sigma^+_{\mathfrak{S}}$. In case of the union rule we have $X \to U \in \Sigma_{j-1}$ and $X \to W \in \Sigma_{j-1}$ with $Y = UW$ and $unique(X) \notin \Sigma^+_{\mathfrak{S}}$. In case of the null transitivity rule we know that there are $X \to Z$ and $Z \to Y$ in Σ_{j-1} with $Z \subseteq XT_s$. If $X \to Z$ is non-trivial, then we are done. If $X \to Z$ is trivial, then $Z \to Y$ is non-trivial since otherwise $X \to Y$ would be trivial, too. If $unique(Z) \in \Sigma^+_{\mathfrak{S}}$, then an application of the null pullback rule to $unique(Z) \in \Sigma^+_{\mathfrak{S}}$, $X \to Z \in \Sigma^+_{\mathfrak{S}}$ and $Z \subseteq XT_s$ shows that $unique(X) \in \Sigma^+_{\mathfrak{S}}$ holds as well. This is a contradiction, i.e., $unique(Z) \notin \Sigma^+_{\mathfrak{S}}$. We have just shown that there is some non-trivial FD $X' \to Y' \in \Sigma$ such that $unique(X') \notin \Sigma^+_{\mathfrak{S}}$.

The equivalence between conditions *2.* and *3* follows immediately from Lemma 1. □

The following result follows directly from Theorem 5 and Theorem 2.

Theorem 6. *The problem whether an SQL table definition T is in Boyce-Codd-Heath Normal Form with respect to an NFS T_s and a set Σ of UCs and FDs over T can be decided in $\mathcal{O}(||\Sigma|| \times |\Sigma|)$ time.* □

If we define a primary key for an SQL table definition, i.e., there is some $X \subseteq T$ such that $X \subseteq T_s$ and $unique(X) \in \Sigma$, then the BCHNF condition for SQL table definitions reduces to the BCHNF condition for relation schemata: T is in BCHNF with respect to Σ and T_s if and only if for all non-trivial FDs $X \to Y \in \Sigma$ we have $X \to T \in \Sigma^+_{\mathfrak{S}}$. However, the presence of primary keys does not mean that the decomposition or synthesis approach [6,16,19] can eliminate any data redundancy.

Example 4. Consider the SQL table definition $T = \{Address, City, ZIP\}$ with $T_s = \{Address, ZIP\}$ and

$$\Sigma = \{unique(Address, City), unique(Address,\ ZIP), ZIP \to City\}.$$

Hence, we have a primary key $\{Address,\ ZIP\}$. A synthesis into the following table definitions:

- $\{City, ZIP\}$ with $ZIP \rightarrow City$ and NFS $\{ZIP\}$,
- $\{Address, City\}$ with $unique(Address, City)$ and NFS $\{Address\}$, and
- $\{Address, ZIP\}$ with $unique(Address, ZIP)$ and NFS $\{Address, ZIP\}$

is dependency-preserving, but neither lossless nor is the first resulting table definition in BCHNF. The tables

Address	*City*	*ZIP*
03 Hudson St	`ni`	10001
70 King St	`ni`	10001

City	*ZIP*
`ni`	10001
`ni`	10001

Address	*ZIP*
03 Hudson St	10001
70 King St	10001

Address	*City*
03 Hudson St	`ni`
70 King St	`ni`

show the synthesis on the semantic level. For this example, it appears to be sensible to replace the FD $ZIP \rightarrow City$ on $\{City, ZIP\}$ by the UC $unique(ZIP)$. The resulting synthesis would then be lossless, all schemata would be in BCHNF and the second table from the left would only contain one row. □

The example illustrates that the approaches of synthesis and decomposition to database normalization require new attention when we consider the features of SQL that allow duplicate and partial information. The presence of duplicates requires uniqueness constraints in addition to functional dependencies, but uniqueness constraints are not preserved when performing joins. Hence, it is not clear what *dependency-preservation* means. The presence of null values requires join attributes to be `NOT NULL` when *lossless* decompositions are to be achieved. Furthermore, projection becomes more difficult to define when duplicates are to be eliminated only sometimes.

8 Conclusion

The class of uniqueness constraints is not subsumed by the class of functional dependencies over SQL tables, in contrast to relations. For this purpose, we have characterized the implication problem for the combined class of UCs and FDs in the presence of `NOT NULL` constraints axiomatically, algorithmically and logically. We have further proposed a syntactic Boyce-Codd-Heath normal form condition for SQL table definitions, and justified this condition semantically. That is, the condition characterizes the absence of redundant data value occurrences in all possible SQL tables. On one hand, the semantics of SQL really calls for a comprehensive support to specify and maintain FDs to guarantee consistency and locate data redundancy. On the other hand, the SQL features motivate a thorough study of the decomposition and synthesis approaches towards achieving normalization.

Acknowledgement

This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand. The second author is supported by a research grant of the Alfried Krupp von Bohlen and Halbach foundation, administered by the German Scholars organization.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Trans. Database Syst. 29(1), 195–232 (2004)
3. Arenas, M., Libkin, L.: An information-theoretic approach to normal forms for relational and XML data. J. ACM 52(2), 246–283 (2005)
4. Armstrong, W.W.: Dependency structures of database relationships. Information Processing 74, 580–583 (1974)
5. Atzeni, P., Morfuni, N.: Functional dependencies and constraints on null values in database relations. Information and Control 70(1), 1–31 (1986)
6. Biskup, J., Dayal, U., Bernstein, P.: Synthesizing independent database schemas. In: SIGMOD Conference, pp. 143–151 (1979)
7. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
8. Codd, E.F.: Recent investigations in relational data base systems. In: IFIP Congress, pp. 1017–1021 (1974)
9. Date, C., Darwen, H.: A guide to the SQL standard. Addison-Wesley Professional, Reading (1997)
10. Diederich, J., Milton, J.: New methods and fast algorithms for database normalization. ACM Trans. Database Syst. 13(3), 339–365 (1988)
11. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
12. Hartmann, S., Link, S.: Numerical constraints on XML data. Inf. Comput. 208(5), 521–544 (2010)
13. Hartmann, S., Link, S.: When data dependencies over SQL tables meet the Logics of Paradox and S-3. In: PODS Conference (2010)
14. Heath, I.J.: Unacceptable file operations in a relational data base. In: SIGFIDET Workshop, pp. 19–33 (1971)
15. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. J. ACM 31(4), 761–791 (1984)
16. Köhler, H.: Finding faithful Boyce-Codd normal form decompositions. In: Cheng, S.-W., Poon, C.K. (eds.) AAIM 2006. LNCS, vol. 4041, pp. 102–113. Springer, Heidelberg (2006)
17. Köhler, H., Link, S.: Armstrong axioms and Boyce-Codd-Heath normal form under bag semantics. Inf. Process. Lett. 110(16), 717–724 (2010)
18. Lien, E.: On the equivalence of database models. J. ACM 29(2), 333–362 (1982)
19. Makowsky, J.A., Ravve, E.V.: Dependency preserving refinements and the fundamental problem of database design. Data Knowl. Eng. 24(3), 277–312 (1998)
20. Sagiv, Y., Delobel, C., Parker Jr., D.S., Fagin, R.: An equivalence between relational database dependencies and a fragment of propositional logic. J. ACM 28(3), 435–453 (1981)
21. Schaerf, M., Cadoli, M.: Tractable reasoning via approximation. Artif. Intell. 74, 249–310 (1995)
22. Vincent, M.: Semantic foundation of 4NF in relational database design. Acta Inf. 36, 1–41 (1999)
23. Vincent, M., Liu, J., Liu, C.: Strong FDs and their application to normal forms in XML. ACM Trans. Database Syst. 29(3), 445–462 (2004)
24. Zaniolo, C.: Database relations with null values. J. Comput. Syst. Sci. 28(1), 142–166 (1984)

Hybrid Logics and NP Graph Properties

Francicleber Martins Ferreira[1,⋆], Cibele Matos Freire[1],
Mario R.F. Benevides[2,⋆⋆], L. Menasché Schechter[3],
and Ana Teresa Martins[1,⋆⋆⋆]

[1] Federal University of Ceará, Computer Science Department
[2] Federal University of Rio de Janeiro, COPPE/Systems
[3] Federal University of Rio de Janeiro, Department of Computer Science

Abstract. We show that for each property of graphs $\mathcal{G}$ in NP there is a sequence $\phi_1, \phi_2, \ldots$ of formulas of the full hybrid logic which are satisfied exactly by the frames in $\mathcal{G}$. Moreover, the size of ϕ_n is bounded by a polynomial. We also show that the same holds for each graph property in the polynomial hierarchy.

1 Introduction

The use of graphs as a mathematical abstraction of objects and structures makes it one of the most used concepts in computer science. Several typical problems in computer science have their inputs modeled by graphs, and such problems commonly involve evaluating some graph property. To mention a well-known example, deciding whether a map can be colored with a certain number of colors is related to a similar problem on planar graphs [10,14]. The applications of graphs in computer science are not restricted to modelling the input of problems. Graphs can be used in the theoretical framework in which some branches of computer science are formalized. This is the case, for example, in distributed systems, in which the model of computation is built on top of a graph [3,12]. Again, properties of graphs can be exploited in order to obtain results about such models of computation.

We can use logic to express properties of structures like graphs. From the semantical standpoint, a logic can be regarded as a pair $\mathcal{L} = (L, \models)$, where L, the language of $\mathcal{L}$, is a set of elements called formulas, and $\models$ is a binary satisfaction relation between some set of objects or structures and formulas. Such set of structures can be a set of relational structures, for example, the class of graphs. A sentence ϕ of L can be used to express some property of structures, hence one could check whether a structure $\mathfrak{A}$ has some property by evaluating whether $\mathfrak{A} \models \phi$ holds or not. The problem of checking whether a given model satisfies a given formula is called model checking.

[⋆] This research was partially supported by CAPES(DS,PROCAD 2010).

[⋆⋆] This research was partially supported by FAPERJ, CNPq and CAPES.

[⋆⋆⋆] This research was partially supported by CNPq(PQ, Universal 2010, Casadinho 2008) and CAPES(PROCAD 2009).

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 123–134, 2011.

In the last few decades, modal logics have attracted the attention of computer scientists working with logic and computation [5]. Among the reasons is the fact that modal logics often have interesting computer theoretical properties, like decidability [15,9]. This is due to a lack of expressive power in comparison with other logics such as first-order logics and its extensions. Many modal logics present also good logical properties, like interpolation, definability, and so on. Research in modal logic includes augmenting the expressive power of the logic using resources as fixed-point operators [6] or hybrid languages [2,1] and keep the nice properties of the logic. Modal logics are particularly suitable to deal with graphs because standard semantics of modal logics is based on structures called *frames*, which are essentially graphs.

In [4], hybrid logics are used to express graph properties, like being connected, hamiltonian or eulerian. Several hybrid logics and fragments were studied to define graph properties through the concept of *validity in a frame* (see Definition 5 below). Some graph properties, like being hamiltonian, require a high expressive power and cannot be expressed by a single sentence in traditional hybrid logics. There are, however, sentences ϕ_n which can express such properties for frames of size n.

We are interested in expressing graph properties in NP (that is, to decide whether a graph belongs to the property is NP) using hybrid logics. The Hybrid Modal logics that we studied have low expressive power in comparison, for example, to second-order logic, hence we do not aim to associate to each graph property a single formula. Instead, we present, to each graph property, a sequence of hybrid sentences ϕ_1, ϕ_2, $\ldots$, such that a graph of size n has the desired property iff ϕ_n is valid in the graph, regarded as a frame. In Section 2, we define the hybrid logic which we will study In Section 3, we show that, for any graph property, there is a sequence of sentences ϕ_1, ϕ_2, $\ldots$ of the fragment of hybrid logic with nominals and the @ operator such that a graph of size n has the property iff ϕ_n is valid in the corresponding frame. However, the size of ϕ_n obtained is exponential on n. In Section 4, we show that, for graph properties in NP, and more generally in the polynomial hierarchy, there is such a sequence, but the size of the sentences is bounded by a polynomial on n. In Section 5, we show how to obtain the results of the previous section for the fragment of hybrid logic without the global modality E and without nominals, provided that graphs are connected.

2 Hybrid Logic

In this section, we present the hybrid logic and its fragments which we will use. Hybrid modal logics extend classical modal logics by adding nominals and state variables to the language. Nominals and state variables behave like propositional atoms which are true in exactly one world. Other extensions include the operators $\downarrow$ (binder) and @. The $\downarrow$ allows one to assign the current state to a state variable. This can be used to keep a record of the visited states. The @ operator allows one to evaluate a formula in the state assigned to a certain nominal or state variable.

Definition 1. *The language of the* hybrid graph logic with the $\downarrow$ binder *is a hybrid language consisting of a set* $PROP$ *of countably many proposition symbols* $p_1, p_2, \ldots$, *a set* NOM *of countably many nominals* $i_1, i_2, \ldots$, *a set* $\mathcal{S}$ *of countably many state-variables* $x_1, x_2, \ldots$, *such that* $PROP$, NOM *and* $\mathcal{S}$ *are pairwise disjoint, the boolean connectives* $\neg$ *and* $\wedge$ *and the modal operators* $@_i$, *for each nominal* i, $@_x$, *for each state-variable* x, $\Diamond$, $\Diamond^{-1}$ *and* $\downarrow$. *The language* L_{FHL} *of the (Full) Hybrid Logic can be defined by the following BNF rule:*

$$\alpha := p \mid t \mid \neg\alpha \mid \alpha \wedge \alpha \mid \Diamond\alpha \mid \Diamond^{-1}\alpha \mid E\alpha \mid @_t\alpha \mid \ \downarrow x.\alpha \mid \top,$$

where t *is either a nominal or a state variable. For each* $C \subseteq \{@, \downarrow, \Diamond^{-1}, E\}$, *we define* $HL(C)$ *to be the corresponding fragment. In particular, we define* $FHL = HL(@, \downarrow, \Diamond^{-1}, E)$. *We also use* $HL(C)\backslash NOM$ *and* $HL(C)\backslash PROP$ *to refer to the fragments of* $HL(C)$ *without nominals and propositional symbols respectively.*

The standard boolean abbreviations $\rightarrow$, $\leftrightarrow$, $\vee$ and $\perp$ can be used with the standard meaning as well as the abbreviations of the dual modal operators: $\Box\phi := \neg\Diamond\neg\phi$, $\Box^{-1}\phi := \neg\Diamond^{-1}\neg\phi$ and $A\phi := \neg E\neg\phi$.

Formulas of hybrid modal logics are evaluated in *hybrid Kripke structures* (or *hybrid models*). These structures are built from *frames.*

Definition 2. *A* frame *is a graph* $\mathcal{F} = (W, R)$, *where* W *is a non-empty set (finite or not) of vertices and* R *is a binary relation over* W, *i.e.,* $R \subseteq W \times W$.

Definition 3. *A* (hybrid) model *for the hybrid logic is a pair* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$, *where* $\mathcal{F}$ *is a frame and* $\mathbf{V} : PROP \cup NOM \mapsto \mathcal{P}(W)$ *is a valuation function mapping proposition symbols into subsets of* W, *and mapping nominals into singleton subsets of* W, *i.e, if* i *is a nominal then* $\mathbf{V}(i) = \{v\}$ *for some* $v \in W$.

In order to deal with the state-variables, we need to introduce the notion of *assignments.*

Definition 4. *An* assignment *is a function* g *that maps state-variables to vertices of the model,* $g : \mathcal{S} \mapsto W$. *We use the notation* $g' = g[v_1/x_1, \ldots, v_n/x_n]$ *to denote an assignment such that* $g'(x) = g(x)$ *if* $x \notin \{x_1, \ldots, x_n\}$ *and* $g'(x_i) = v_i$, *otherwise.*

The semantical notion of satisfaction is defined as follows:

Definition 5. *Let* $\mathcal{M} = (\mathcal{F}, \mathbf{V})$ *be a model. The notion of* satisfaction *of a formula* φ *in a model* $\mathcal{M}$ *at a vertex* v *with assignment* g, *notation* $\mathcal{M}, g, v \Vdash \varphi$, *can be inductively defined as follows:*
$\mathcal{M}, g, v \Vdash p$ *iff* $v \in \mathbf{V}(p)$;
$\mathcal{M}, g, v \Vdash \top$ *always;*
$\mathcal{M}, g, v \Vdash \neg\varphi$ *iff* $\mathcal{M}, g, v \nVdash \varphi$;
$\mathcal{M}, g, v \Vdash \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{M}, g, v \Vdash \varphi_1$ *and* $\mathcal{M}, g, v \Vdash \varphi_2$;
$\mathcal{M}, g, v \Vdash \Diamond\varphi$ *iff there is a* $w \in W$ *such that* vRw *and* $\mathcal{M}, g, w \Vdash \varphi$;
$\mathcal{M}, g, v \Vdash \Diamond^{-1}\varphi$ *iff there is a* $w \in W$ *such that* wRv *and* $\mathcal{M}, g, w \Vdash \varphi$;

$\mathcal{M}, g, v \Vdash i$ *iff* $v \in \mathbf{V}(i)$*;*
$\mathcal{M}, g, v \Vdash @_i\varphi$ *iff* $\mathcal{M}, g, d_i \Vdash \varphi$*, where* $d_i \in \mathbf{V}(i)$*;*
$\mathcal{M}, g, v \Vdash x$ *iff* $g(x) = v$*;*
$\mathcal{M}, g, v \Vdash @_x\phi$ *iff* $\mathcal{M}, g, d \Vdash \phi$*, where* $d = g(x)$*;*
$\mathcal{M}, g, v \Vdash \downarrow x.\phi$ *iff* $\mathcal{M}, g[v/x], v \Vdash \phi$*.*

For each nominal i, the formula $@_i\varphi$ means that if $\mathbf{V}(i) = \{v\}$ then φ is satisfied at v. If $\mathcal{M}, g, v \Vdash \varphi$ for every vertex v, we say that φ is *globally satisfied* in the model $\mathcal{M}$ with assignment g ($\mathcal{M}, g \Vdash \varphi$) and if φ is globally satisfied in all models $\mathcal{M}$ and assignments of a frame $\mathcal{F}$, we say that φ is *valid* in $\mathcal{F}$ ($\mathcal{F} \Vdash \varphi$).

3 Properties of Graphs in HL

In [4], it was shown that there is a formula ϕ_n of FHL such that a graph of size n is Hamiltonian iff it globally satisfies ϕ_n. The main question which underlies this investigation is whether there is a sequence of formulas $(\phi_n)_{n\in N}$ for each graph property $\mathcal{G}$ in NP such that a graph G of size n is in $\mathcal{G}$ iff G, as a frame, globally satisfies ϕ_n. Actually, we can show that such sequence exists for each graph property.

Let $G = (V, E)$ be a graph of cardinality n. Let us consider that the set V of vertices coincides with the set $\{1, \ldots, n\}$ of nominals. Consider the formula:

$$\psi_G = \bigwedge_{(i,j)\in E} @_i\Diamond j \wedge \bigwedge_{(i,j)\notin E} @_i\neg\Diamond j.$$

Let $\mathcal{G}$ be any property of graphs. We define the formulas

$$\psi_{\mathcal{G}}^n = \bigvee_{G\in\mathcal{G},|G|=n} \psi_G \quad , \quad \theta^n = \bigwedge_{i,j\in\{1,\ldots,n\},i\neq j} @_i\neg j \quad \text{and} \quad \phi_{\mathcal{G}}^n = \theta^n \rightarrow \psi_{\mathcal{G}}^n.$$

Lemma 1. *Let G be a graph of cardinality n and $\mathcal{G}$ a property of graphs. Then $G \in \mathcal{G}$ iff $G \Vdash \phi_{\mathcal{G}}^n$.*

Since there are 2^{n^2} graphs with vertices in $\{1, \ldots, n\}$, we have that the size of $\phi_{\mathcal{G}}^n$ is $O(2^{n^2})$ for any graph property $\mathcal{G}$. Obviously, there is no hope for that sequence of formulas to be always computable. We can show, however, that, for problems in the polynomial hierarchy, such sequence is recursive and, moreover, there is a polynomial bound in the size of formulas.

4 Translation

In this section, we show that for each graph property $\mathcal{G}$ in the polynomial hierarchy there is a sequence $(\phi_n)_{n\in N}$ of formulas such that a graph G of size n is in $\mathcal{G}$ iff $G \Vdash \phi_n$ and such that ϕ_n is bounded from above by a polynomial on n. We will use the well-known characterization of problems in the polynomial hierarchy and classes of finite models definable in second-order logic (SO) from descriptive complexity theory [11]. To this end, we define a translation from formulas in SO to formulas in FHL which are equivalent with respect to frames of size n, for some $n \in N$. Such translation will give us formulas whose size is bounded by a polynomial on n. Moreover, the formulas obtained by the translation do not use propositional symbols, nominals or free state variables, which means that, for these formulas, the complexity of model-checking and frame-checking coincides. We use the well known definitions and concepts related to first-order logic (FO) and second-order logic which can be founded in most textbooks (see, for instance, [7]).

Definition 6 (Translation from FO to HL). *Let ϕ be a first-order formula in the vocabulary $S = \{E, R_1, \ldots, R_m\}$ where E is binary, n a natural number and f a function from the set of first-order variables into $\{1, \ldots, n\}$. Let $t, z_1, \ldots, z_n$ be state variables and for each $R \in \{R_1, \ldots, R_m\}$ of arity h, let $y^R_{j_1,\ldots,j_h}$ be a state variable, with $j_i \in \{1, \ldots, n\}$, $1 \leq i \leq h$. We define the function $tr^f_n : L^S_{FO} \to L_{FHL}$ as:*

- $tr^f_n(x_1 \equiv x_2) = @_{z_{f(x_1)}} z_{f(x_2)}$;
- $tr^f_n(E(x_1, x_2)) = @_{z_{f(x_1)}} \Diamond z_{f(x_2)}$;
- $tr^f_n(R(x_1, \ldots, x_k)) = @_t y^R_{f(x_1),\ldots,f(x_k)}$, *for each* $R \in \{R_1, \ldots, R_m\}$;
- $tr^f_n(\gamma \wedge \theta) = tr^f_n(\gamma) \wedge tr^f_n(\theta)$;
- $tr^f_n(\neg\gamma) = \neg tr^f_n(\gamma)$;
- $tr^f_n(\exists x\gamma) = \bigvee_{i=1}^n tr^{f\frac{x}{i}}_n(\gamma)$;
- $tr^f_n(\forall x\gamma) = \bigwedge_{i=1}^n tr^{f\frac{x}{i}}_n(\gamma)$.

In the translation above, t represents a state v such that, if $z^R_{j_1,\ldots,j_h}$ is assigned to v and $z_{j_1}, \ldots, z_{j_h}$ are assigned to $v_1, \ldots, v_h$, then $(v_1, \ldots, v_h)$ belongs to the interpretation of R. The function $f\frac{x}{i}$ maps x to i and y to $f(y)$ for $y \neq x$. The translation above only works for frames with more than one state, but, since there are only two frames of size 1, we can state for each graph property which frames of size 1 belong to the property.

Note that if ϕ is a sentence, then $tr^f_n(\phi) = tr^{f'}_n(\phi)$. Hence we write $tr_n(\phi)$ instead of $tr^f_n(\phi)$ for a sentence ϕ.

Example 1. We give an example of application of the translation above. Let $\oplus(\phi, \psi, \theta)$ be the ternary exclusive "or". Consider the following first-order sentence:

$$\phi := \forall x\big(\oplus (R(x), G(x), B(x))\big) \wedge \forall x\forall y\big((E(x,y) \wedge x \neq y) \to \\ \neg((R(x) \wedge R(y)) \vee (G(x) \wedge G(y)) \vee (B(x) \wedge B(y)))\big).$$

The sentence above says that each element belongs to one of the sets R, G and B, each adjacent pair does not belong to the same set, and no element belongs to more than one set. This sentence is true iff the sets R, G and B forms a 3-coloring of a graph with edges in E. Below we translate ϕ into a formula of hybrid logic using the translation given above and setting $n = 3$:

$$tr_3(\phi) := \bigwedge_{i=1}^{3} \left(\oplus \left(@_t y_i^R, @_t y_i^G, @_t y_i^B\right)\right) \wedge \bigwedge_{i=1}^{3} \left[\bigwedge_{j=1}^{3} \left((@_{z_i} \Diamond z_j \wedge \neg @_{z_i} z_j\right) \rightarrow \right.$$

$$\left. \neg((@_t y_i^R \wedge @_t y_j^R) \vee (@_t y_i^G \wedge @_t y_j^G) \vee (@_t y_i^B \wedge @_t y_j^B))) \right].$$

Lemma 2. *$tr_n(\phi)$ has polynomial size in n for a fixed ϕ, that is, $tr_n(\phi) \in O(n^k)$ for some $0 \leq k$.*

Proof. By induction on ϕ one can see that $tr_n(\phi)$ is $O(n^k)$, where k is the quantifier rank of ϕ, that is, the maximum number of nested quantifiers.

Lemma 3. *Let $G = (V, E^G)$ be a graph of cardinality n, $\mathbf{R}_1, \ldots, \mathbf{R}_m$ relations on V with arities $r_1, \ldots, r_m$, g an assignment of state variables, β an assignment of first-order variables, $S = \{E, R_1, \ldots, R_m\}$ a vocabulary and f a function from the set of first-order variables to $\{1, \ldots, n\}$ such that:*

(i) g assigns to each variable z_i a different element in V;
(ii) $g(y^R_{i_1,\ldots,i_k}) = g(t)$ iff $(g(z_{i_1}), \ldots, g(z_{i_k})) \in \mathbf{R}$ for each $R \in \{R_1, \ldots, R_m\}$;
(iii) $\beta(x) = g(z_{f(x)})$ for each first-order variable x.

If ϕ is a first-order formula in the vocabulary S, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff for all $w \in V$, $(G, g, w) \Vdash tr_n^f(\phi)$.

Definition 7 (Translation from SO to FHL). *Let $\phi = Q_1 X_1 \ldots Q_l X_l \psi$ be a SO formula where $Q_i \in \{\exists, \forall\}$ and ψ is a first-order sentence. We define*

$$T^n(\phi) = \spadesuit_1 \downarrow y^{X_1}_{\overline{1}} \ldots . \spadesuit_1 \downarrow y^{X_1}_{\overline{n}} \ldots . \spadesuit_l \downarrow y^{X_l}_{\overline{1}} \ldots . \spadesuit_l \downarrow y^{X_l}_{\overline{n}} tr_n(\psi),$$

where $\spadesuit_i = E$ if $Q_i = \exists$ and A otherwise.

Example 2. Consider the sentence ϕ of Example 1. Let ψ be the following second-order sentence:

$$\psi := \exists R \exists G \exists B(\phi).$$

The sentence ψ above states that there are three sets R, G and B which forms a 3-coloring of elements in the domain of a structure. Hence, ϕ is satisfied in a graph with edges in E iff such graph is 3-colorable. Deciding whether a graph is 3-colorable is a NP-complete problem [13]. We apply the translation T^n for $n = 3$ below. Let

$$\hat{\mathbf{Q}} := E \downarrow y_1^R.E \downarrow y_2^R.E \downarrow y_3^R.E \downarrow y_1^G.E \downarrow y_2^G.E \downarrow y_3^G.E \downarrow y_1^B.E \downarrow y_2^B.E \downarrow y_3^B..$$

We have $T^3(\psi) := \hat{\mathbf{Q}} tr_3(\phi)$. That is,

$$T^3(\phi) := \hat{\mathbf{Q}} \left(\bigwedge_{i=1}^{3} \left(\oplus \left(@_t y_i^R, @_t y_i^G, @_t y_i^B\right)\right) \wedge \bigwedge_{i=1}^{3} \left[\bigwedge_{j=1}^{3} \left((@_{z_i} \Diamond z_j \wedge \neg @_{z_i} z_j\right) \rightarrow \right.\right.$$

$$\left.\left. \neg((@_t y_i^R \wedge @_t y_j^R) \vee (@_t y_i^G \wedge @_t y_j^G) \vee (@_t y_i^B \wedge @_t y_j^B))) \right] \right).$$

Lemma 4. *Let $G = (V, E^G)$ be a graph of cardinality n, $\mathbf{R}_1, \dots, \mathbf{R}_m$ relations on V with arities $r_1, \dots, r_m$, g an assignment of state variables, β an assignment of first-order variables, $S = \{E, R_1, \dots, R_m\}$ a vocabulary and f a function from the set of first-order variables to $\{1, \dots, n\}$ such that:*

(i) g assigns to each variable z_i a different element in V;

(ii) $g(y^R_{i_1,\dots,i_k}) = g(t)$ iff $(g(z_{i_1}), \dots, g(z_{i_k})) \in \mathbf{R}$ for each $R \in \{R_1, \dots, R_m\}$;

(iii) $\beta(x) = g(z_{f(x)})$ for each first-order variable x.

If $\phi = Q_1X_1 \dots Q_lX_l\psi$ is a second-order formula in the symbol set S, then $(G, \mathbf{R}_1, \dots, \mathbf{R}_m, \beta) \models \phi$ iff for all $w \in V$, $(G, g, w) \Vdash T^n(\phi)$.

We have the following:

Theorem 1. *Let ϕ be a second-order sentence and G a graph of cardinality n. Then $G \models \phi$ iff*

$$G \Vdash \downarrow t.E \downarrow z_1 \dots E \downarrow z_n. \left(\bigwedge_{1 \le i < j \le n} @_{z_i} \neg z_j \wedge T^n(\phi) \right).$$

A well known result of descriptive complexity is the correspondence between the polynomial hierarchy and the alternation hierarchy of second-order logic (with respect to finite models) [11,8].

There are several ways to define this hierarchy, for example using alternating Turing machines [11]. In this paper we assume the definition presented in [13], which uses Turing machines with oracles to define PH.

A Turing machine with an oracle is a machine that has the special ability of guessing some specific questions. When a Turing machine has an oracle for a decision problem B, during its execution it can ask for the oracle if some instance of problem B is positive or negative. This is done in constant time, regardless the size of the instance. We use the notation M^B to define a Turing machine M with an oracle for a problem B. In a similar way, we define $\mathcal{C}^{\mathcal{B}}$, where $\mathcal{C}$ and $\mathcal{B}$ are complexity classes, as the class of problems solved by a Turing machine in $\mathcal{C}$ with an oracle in $\mathcal{B}$.

Definition 8. *Consider the following sequence of complexity classes. First, $\Delta^p_0 = \Sigma^p_0 = \Pi^p_0 = PTIME$ and, for all $i \geq 0$,*

1. $\Delta^p_{i+1} = P^{\Sigma^p_i}$
2. $\Sigma^p_{i+1} = NP^{\Sigma^p_i}$
3. $\Pi^p_{i+1} = coNP^{\Sigma^p_i}$.

We define the Polynomial Time Hierarchy as the class $PH = \bigcup_{i \geq 0} \Sigma^p_i$.

In particular we have:

Theorem 2 ([11]). *Let $\mathcal{G}$ be a graph property in the polynomial hierarchy. Then there is a second-order sentence ϕ in the language of graphs such that $G \in \mathcal{G}$ iff $G \models \phi$.*

The following is the main theorem of this section:

Theorem 3. *Let $\mathcal{G}$ be a graph property in the polynomial hierarchy. Then there is a set of sentences $\Phi = \{\phi_1, \phi_2, \ldots\}$ of FHL, such that:*
(1) $G \in \mathcal{G}$ iff $G \Vdash \Phi$ iff $G \Vdash \phi_{|G|}$, and
(2) $|\phi_n|$ is $O(n^k)$ for some constant k depending only on $\mathcal{G}$.

Corollary 1. *If $\phi \in \exists SO$, the existencial fragment of SO, then T^n is in HL$(@, \downarrow, E) \backslash \{\downarrow \Box \downarrow, PROP\}$, that is, the fragment of HL$(@, \downarrow, E)$ without propositional symbols and the pattern $\downarrow \Box \downarrow$ (which means an $\downarrow$ into the scope of a $\Box$ which in your turn is into the scope of an $\downarrow$).*

5 Connected Frames with Loops

Let FHL$\backslash\{E, NOM\}$ be the fragment of full hybrid logic without the modality E and without nominals. Its is not difficult to show that:

Lemma 5. *Frame validity and model (global) satisfaction for sentences from FHL$\backslash\{E, NOM\}$ are invariant under disjoint union.*

Thus an analogous to Theorem 3 does not hold for FHL$\backslash\{E, NOM\}$.

Corollary 2. *There are graph properties in PTIME for which there is no set $\Phi = \{\phi_1, \phi 2, \ldots\}$ from sentences in FHL$\backslash\{E, NOM\}$ which satisfies conditions (1) and (2) from Theorem 3 above.*

Proof. Connectivity is one such a property.

However, Theorem 3 still hold if we restrict ourselves to connected, frames with loops. Consider the following translation from SO to FHL$\backslash\{E, NOM\}$:

Definition 9. *Let $\phi = Q_1 X_1 \ldots Q_l X_l \psi$ be a second-order formula where $Q_i \in \{\exists, \forall\}$ and ψ is a first-order sentence. We define*

$$\hat{T}^n(\phi) = \spadesuit_1 \downarrow y_{\overline{1}}^{X_1} \ldots . \spadesuit_1 \downarrow y_{\overline{n}}^{X_1} \ldots . \spadesuit_l \downarrow y_{\overline{1}}^{X_l} \ldots . \spadesuit_l \downarrow y_{\overline{n}}^{X_l} tr_n(\psi),$$

where $\spadesuit_i = (\Diamond\Diamond^{-1})^n$ if $Q_i = \exists$ and $(\Box\Box^{-1})^n$ otherwise.

Lemma 6. *Let $G = (V, E^G)$ be a connected graph with loops on each vertex, $\mathbf{R}_1, \ldots, \mathbf{R}_m$ relations on V with arities $r_1, \ldots, r_m$, g an assignment of state variables, β an assignment of first-order variables, $S = \{E, R_1, \ldots, R_m\}$ a vocabulary and f a function from the set of first-order variables to $\{1, \ldots, n\}$ such that:*
(i) g assigns to each variable z_i a different element in V;
(ii) $g(y^R_{i_1,\ldots,i_k}) = g(t)$ iff $(g(z_{i_1}), \ldots, g(z_{i_k})) \in \mathbf{R}$ for each $R \in \{R_1, \ldots, R_m\}$;
(iii) $\beta(x) = g(z_{f(x)})$ for each first-order variable x.
If $\phi = Q_1 X_1 \ldots Q_l X_l \psi$ is a second-order formula in the symbol set S, then

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi \text{ iff for all } w \in V,\ (G, g, w) \Vdash \hat{T}^n(\phi).$$

Proof. Analogous to the proof of Lemma 4

Theorem 4. *Let ϕ be a second-order sentence and G a connected graph of cardinality n with loops. Then $G \models \phi$ iff*

$$G \Vdash \downarrow t.(\Diamond\Diamond^{-1})^n \downarrow z_1.\ldots(\Diamond\Diamond^{-1})^n \downarrow z_n.\left(\bigwedge_{1\leq i<j\leq n} @_{z_i}\neg z_j \wedge \hat{T}^n(\phi)\right).$$

Proof. Analogous to the proof of Theorem 1.

Theorem 5. *Let $\mathcal{G}$ be a property of connected graphs with loops in the polynomial hierarchy. There is a set of sentences $\Phi = \{\phi_1, \phi_2, \ldots\}$ of the fragment $FHL\backslash\{E, NOM\}$, such that:*

(1)for all connected graphs G with loops, $G \in \mathcal{G}$ iff $G \Vdash \Phi$ iff $G \Vdash \phi_{|G|}$, and

(2) $|\phi_n|$ is $O(n^k)$ for some constant k depending only on $\mathcal{G}$.

References

1. Areces, C., Blackburn, P., Marx, M.: Hybrid logics: Characterization, interpolation and complexity. Journal of Symbolic Logic 66(3), 977–1010 (2001)
2. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, vol. 3, pp. 821–868. Elsevier Science Ltd., Amsterdam (2007)
3. Barbosa, V.: An introduction to distributed algorithms. The MIT Press, Cambridge (1996)
4. Benevides, M., Schechter, L.: Using modal logics to express and check global graph properties. Logic Journal of IGPL 17(5), 559 (2009)
5. Blackburn, P., De Rijke, M., Venema, Y.: Modal logic. Cambridge Univ. Pr., Cambridge (2002)
6. Bradfield, J., Stirling, C.: Modal mu-calculi. Handbook of Modal Logic 3, 721–756 (2007)
7. Ebbinghaus, H., Flum, J., Thomas, W.: Mathematical logic. Springer, Heidelberg (1994)
8. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R. (ed.) Complexity of Computation. SIAM-AMS Proceedings, vol. 7, pp. 43–73. AMS, Providence (1974)
9. Grädel, E.: Why are modal logics so robustly decidable? In: Paun, G., Rozenberg, G., Salomaa, A. (eds.) Current Trends in Theoretical Computer Science. Entering the 21st Century, pp. 393–408. World Scientific, Singapore (2001)
10. Haken, W., Appel, K., Koch, J.: Every planar map is four colorable. Contemporary Mathematics, vol. 98. American Mathematical Society, Providence (1989)
11. Immerman, N.: Descriptive complexity. Springer, Heidelberg (1999)
12. Lynch, N.: Distributed algorithms. Morgan Kaufmann, San Francisco (1996)
13. Papadimitriou, C.: Computational complexity. John Wiley and Sons Ltd., Chichester (2003)
14. Robertson, N., Sanders, D., Seymour, P., Thomas, R.: The four-colour theorem. Journal of Combinatorial Theory, Series B 70(1), 2–44 (1997)
15. Vardi, M.: Why is modal logic so robustly decidable. In: Immerman, N., Kolaitis, P.G. (eds.) Descriptive Complexity and Finite Models. Discrete Mathematics and Theoretical Computer Science, vol. 31, pp. 149–184. American Mathematical Society, Providence (1996)

A Proofs of Main Theorems

Proof (Proof of Lemma 3). We proceed by induction on ϕ.

- ϕ is an atomic formula: In this case $\phi = x \equiv y$, $\phi = E(x,y)$ or $\phi = R_i(x_1,\ldots,x_n)$. If $\phi = x \equiv y$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $\beta(x) = \beta(y)$ iff, by (iii), $g(z_{f(x)}) = g(z_{f(y)})$ iff $(G, g, w) \Vdash @_{z_{f(x)}} z_{f(y)} = tr_n^f(\phi)$. If $\phi = E(x,y)$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $(\beta(x), \beta(y)) \in E^G$ iff, by (iii), $(g(z_{f(x)}), g(z_{f(y)})) \in E^G$ iff $(G, g, w) \Vdash @_{z_{f(x_1)}} \Diamond z_{f(x_2)} = tr_n^f(\phi)$. If $\phi = R_i(x_1, \ldots, x_h)$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff $(\beta(x_1), \ldots, \beta(x_n)) \in \mathbf{R}_i$ iff, by (iii), $(g(z_{f(x_1)}), \ldots, g(z_{f(x_n)})) \in \mathbf{R}_i$ iff, by (ii), $g(y^R_{f(x_1),\ldots,f(x_k)}) = g(t)$ iff $(G, g, w) \Vdash @_t y^R_{f(x_1),\ldots,f(x_k)} = tr_n^f(\phi)$.
- $\phi = \gamma \wedge \theta$ or $\phi = \neg\gamma$: These cases follow directly from the definition of tr_n^f and the inductive hypothesis.
- $\phi = \exists x \gamma$: In this case, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff there is a $v \in V$ such that $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta\frac{x}{v}) \models \gamma$. By (i), there is a j be such that $v = z_j$. Hence we have $\beta\frac{x}{v}(y) = g(z_{f\frac{x}{j}(y)})$ for each first-order variable y. By inductive hypothesis we have, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta\frac{x}{v}) \models \gamma$ iff $(G, g, w) \Vdash tr_n^{f\frac{x}{j}}(\gamma)$ iff $(G, g, w) \Vdash \bigvee_{i=1}^n tr_n^{f\frac{x}{i}}(\gamma) = tr_n^f(\phi)$.
- $\phi = \forall x \gamma$: It follows that $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff, for each $v \in V$, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta\frac{x}{v}) \models \gamma$. By (i), for each $v \in V$ there is a j such that $v = z_j$. Hence we have $\beta\frac{x}{v}(y) = g(z_{f\frac{x}{j}(y)})$ for each first-order variable y. By inductive hypothesis we have, for each $v \in V$, $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta\frac{x}{v}) \models \gamma$ iff, for each $j \in \{1, \ldots, n\}$, $(G, g, w) \Vdash tr_n^{f\frac{x}{j}}(\gamma)$ iff $(G, g, w) \Vdash \bigwedge_{i=1}^n tr_n^{f\frac{x}{i}}(\gamma) = tr_n^f(\phi)$.

Proof (Proof of Lemma 4). Let $v \in V$ such that $v \neq g(t)$. For each $\mathbf{X}_1$ on V, let $v^{\mathbf{X}_1}_{i_1,\ldots,i_h} = g(t)$ if $(g(z_{i_1}), \ldots, g(z_{i_h}))$ and v otherwise. Then, for each $\mathbf{X}_1$ on V there is an assignment $g^{\mathbf{X}_1}$ defined as

$$g^{\mathbf{X}_1} = g\frac{y^{X_1}_{1,\ldots,1} \cdots y^{X_1}_{i_1,\ldots,i_h} \cdots y^{X_1}_{n,\ldots,n}}{v^{\mathbf{X}_1}_{1,\ldots,1} \cdots v^{\mathbf{X}_1}_{i_1,\ldots,i_h} \cdots v^{\mathbf{X}_1}_{n,\ldots,n}}.$$

On the contrary, given an assignment g' we can find $\mathbf{X}_1$ such that $g' = g^{\mathbf{X}_1}$.

The assignment $g^{\mathbf{X}_1}$ can be described as:

$$g^{\mathbf{X}_1}(s) = \begin{cases} g(t) & \text{, if } s = y^{X_1}_{i_1,\ldots,i_h} \text{ and } (i_1, \ldots, i_h) \in \mathbf{X}_1; \\ v & \text{, for some } v \neq g(t) \text{, if } s = y^{X_1}_{i_1,\ldots,i_h} \text{ and } (i_1, \ldots, i_h) \notin \mathbf{X}_1; \\ g(s) & \text{, otherwise;} \end{cases}$$

It follows that $g^{\mathbf{X}_1}$ and $\mathbf{X}_1$ satisfies (i) and (iii) and

(ii') $g^{\mathbf{X}_1}(y^R_{i_1,\ldots,i_k}) = g^{\mathbf{X}_1}(t)$ iff $(g^{\mathbf{X}_1}(z_{i_1}), \ldots, g^{\mathbf{X}_1}(z_{i_k})) \in \mathbf{R}$,

for each $R \in \{R_1, \ldots, R_m, X_1\}$.

Now, we proceed by induction on the size l of the prefix $Q_1 X_1 \ldots Q_l X_l$. If $l = 0$, then ϕ is first-order and the result follows immediately from Lemma 3.

Suppose that $l > 0$. If $\phi = \exists X_1 \ldots Q_l X_l \psi$, then $(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$ iff there is $\mathbf{X_1} \subseteq V^{r_1}$ such that

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \mathbf{X}_1, \beta) \models Q_2 X_2 \ldots Q_l X_l \psi$$

iff, by the inductive hypothesis, there is $g^{\mathbf{X}_1}$ such that

$$(G, g^{\mathbf{X}_1}, w) \Vdash T^n(Q_2 X_2 \ldots Q_l X_l \psi)$$

iff

$$(G, g\frac{y^{X_1}_{1,\ldots,1} \ldots y^{X_1}_{i_1,\ldots,i_h} \ldots y^{X_1}_{n,\ldots,n}}{v^{\mathbf{X}_1}_{1,\ldots,1} \ldots v^{\mathbf{X}_1}_{i_1,\ldots,i_h} \ldots v^{\mathbf{X}_1}_{n,\ldots,n}}, w) \Vdash T^n(Q_2 X_2 \ldots Q_l X_l \psi)$$

iff

$$(G, g, w) \Vdash E \downarrow y^{X_1}_{\overline{1}} \ldots . E \downarrow y^{X_1}_{\overline{n}} . T^n(Q_2 X_2 \ldots Q_l X_l \psi) = T^n(\phi).$$

If $\phi = \forall X_1 \ldots Q_l X_l \psi$, then

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \beta) \models \phi$$

iff, for all $\mathbf{X_1} \subseteq V^{r_1}$,

$$(G, \mathbf{R}_1, \ldots, \mathbf{R}_m, \mathbf{X}_1, \beta) \models Q_2 X_2 \ldots Q_l X_l \psi$$

iff, by the inductive hypothesis, for all $g^{\mathbf{X}_1}$,

$$(G, g^{\mathbf{X}_1}, w) \Vdash T^n(Q_2 X_2 \ldots Q_l X_l \psi)$$

iff, for all $v^{\mathbf{X}_1}_{1,\ldots,1} \ldots v^{\mathbf{X}_1}_{i_1,\ldots,i_h} \ldots v^{\mathbf{X}_1}_{n,\ldots,n}$,

$$(G, g\frac{y^{X_1}_{1,\ldots,1} \ldots y^{X_1}_{i_1,\ldots,i_h} \ldots y^{X_1}_{n,\ldots,n}}{v^{\mathbf{X}_1}_{1,\ldots,1} \ldots v^{\mathbf{X}_1}_{i_1,\ldots,i_h} \ldots v^{\mathbf{X}_1}_{n,\ldots,n}}, w) \Vdash T^n(Q_2 X_2 \ldots Q_l X_l \psi)$$

iff $(G, g, w) \Vdash A \downarrow y^{X_1}_{\overline{1}} \ldots . A \downarrow y^{X_1}_{\overline{n}} . T^n(Q_2 X_2 \ldots Q_l X_l \psi) = T^n(\phi)$.

Proof (Proof of Theorem 1).

$$(G, g, w) \Vdash \downarrow t . E \downarrow z_1 . \ldots E \downarrow z_n . \left[\left(\bigwedge_{1 \le i < j \le n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right]$$

iff

$$(G, g\frac{t}{w}, w) \Vdash E \downarrow z_1 . \ldots E \downarrow z_n . \left[\left(\bigwedge_{1 \le i < j \le n} @_{z_i} \neg z_j \right) \wedge T^n(\phi) \right]$$

iff there are $v_1, \ldots, v_n \in V$ such that

$$(G, g\frac{t z_1 \ldots z_n}{w v_1 \ldots v_n}, w) \Vdash \left[\left(\bigwedge_{1 \le i < j \le n} @_{z_i} \neg z_j \right) \wedge T^n(\phi) \right]$$

iff there are $v_1 \neq \ldots \neq v_n \in V$ such that

$$(G, g\frac{tz_1 \ldots z_n}{wv_1 \ldots v_n}, w) \Vdash T^n(\phi)$$

iff, by Lemma 4, $G \Vdash \phi$.

Proof (Proof of Theorem 3). Let ψ be a second-order formula expressing $\mathcal{G}$. Let

$$\theta_n = A \downarrow z_1. \ldots A \downarrow z_n. \left[\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \rightarrow A \downarrow z. \left(\bigvee_{1 \leq i \leq n} @_{z_i} z \right) \right].$$

The sentence θ_n says that there are at most n vertices in the frame. We define ϕ_n as:

$$\phi_n = \theta_n \rightarrow \downarrow t.E \downarrow z_1. \ldots E \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right].$$

Let $G \in \mathcal{G}$. Let g be any assignment of state variables and w be any point in G. If $G \nVdash \theta_n$, then $G \Vdash \phi_n$. It follows that $G \Vdash \phi_n$ for each $n \neq |G|$. Hence, $G \Vdash \Phi$ iff $G \Vdash \phi_{|G|}$. Let $|G| = n$. Then $(G, g, w) \Vdash \phi_n$ iff

$$(G, g, w) \Vdash \downarrow t.E \downarrow z_1. \ldots E \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right]$$

iff, by Theorem 1, $G \in \mathcal{G}$.

Proof (Proof of Theorem 5). Let ψ be a second-order formula expressing $\mathcal{G}$. Let $\mathbf{Q}$ be the prefix $(\Box\Box^{-1})^n \downarrow z_1. \ldots (\Box\Box^{-1})^n \downarrow z_n.$ and let

$$\theta_n = \mathbf{Q} \left[\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \rightarrow (\Box\Box^{-1})^n \downarrow z. \left(\bigvee_{1 \leq i \leq n} @_{z_i} z \right) \right], \text{ and}$$

$$\psi_n = \downarrow t.(\Diamond\Diamond^{-1})^n \downarrow z_1. \ldots (\Diamond\Diamond^{-1})^n \downarrow z_n. \left[\left(\bigwedge_{1 \leq i < j \leq n} @_{z_i} \neg z_j \right) \wedge T^n(\psi) \right].$$

Let $\phi_n = \theta_n \rightarrow \psi_n$. The remaining of the proof is similar to the proof of Theorem 2.

On the Expressive Power of IF-Logic with Classical Negation

Santiago Figueira[1,3,⋆], Daniel Gorín[1], and Rafael Grimson[2,3]

[1] Dto. Computación, FCEN, Universidad de Buenos Aires, Argentina
[2] Dto. Matemática, FCEN, Universidad de Buenos Aires, Argentina
[3] CONICET, Argentina

Abstract. It is well-known that Independence Friendly (IF) logic is equivalent to existential second-order logic (Σ^1_1) and, therefore, is not closed under classical negation. The boolean closure of IF sentences, called Extended IF-logic, on the other hand, corresponds to a proper fragment of Δ^1_2. In this paper we consider IF-logic extended with Hodges' flattening operator, which allows classical negation to occur also under the scope of IF quantifiers. We show that, nevertheless, the expressive power of this logic does not go beyond Δ^1_2. As part of the proof, we give a prenex normal form result and introduce a non-trivial syntactic fragment of full second-order logic that we show to be contained in Δ^1_2.

1 Introduction

Independence Friendly logic [4] (IF, for short) is an extension of first-order logic (FO) where each disjunction and each existential quantifier may be decorated with denotations of universally-quantified variables, as in:

$$\forall x \forall y \exists z_{|\forall y} \exists w_{|\forall y} [y = z \vee_{|\forall x, \forall y} w = y] \tag{1}$$

The standard interpretation of IF is through a variation of the classical game-theoretical semantics for FO: Eloïse's strategy function for a position of the form $\exists x_{|\forall y, \forall z} \psi$ or $\psi \vee_{|\forall y, \forall z} \chi$, under valuation v, cannot depend on neither $v(y)$ nor $v(z)$. Thus, we say that a sentence φ is *true* in model $\mathcal{A}$ (notation, $\mathcal{A} \models^{+} \varphi$) if Eloïse has a winning strategy on the associated game; and that it is *false* (notation, $\mathcal{A} \models^{-} \varphi$) whenever Abélard has a winning strategy.

Now, the fact that Eloïse's strategy may not take into account all the available information turns the game into one of "imperfect information". In fact, it is a non-determined game, i.e, for certain instances of the game, no winning strategy exists. As an example of non-determinacy, consider this formula:

$$\chi_1 := \forall x \exists y_{|\forall x} x \neq y \tag{2}$$

It is not hard to see that if $\mathcal{A}$ is a model with at least two elements, then $\mathcal{A} \not\models^{+} \chi_1$ and $\mathcal{A} \not\models^{-} \chi_1$. That is, χ_1 is neither *true* nor *false* in $\mathcal{A}$.

⋆ S. Figueira was partially supported by CONICET Argentina (grant PIP 370).

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 135–145, 2011.

In game-theoretical semantics, negation is interpreted as a switch of roles, i.e., Abélard plays on Eloïse's former positions and vice versa. We use $\sim$ to denote this form of negation and we refer to it as *game negation*. For any IF-formula ψ and any model $\mathcal{A}$, $\mathcal{A} \models^{+} \psi$ iff $\mathcal{A} \models^{-} {\sim}\psi$. (i.e., Eloïse has a winning strategy for ψ on $\mathcal{A}$ iff Abélard has one for ${\sim}\psi$ on $\mathcal{A}$). However, observe that $\psi \vee {\sim}\psi$ is not in general a valid IF-formula (e.g., take ψ to be χ_1 in (2)). This means that game negation in IF is not equivalent to *classical negation*, which will be denoted with $\neg$ and is characterized by:

$$\mathcal{A} \models^{+} \neg\psi \text{ iff } \mathcal{A} \not\models^{+} \psi \tag{3}$$

Since the expressive power of IF corresponds to that of existential second-order logic (Σ^1_1) [4,5] and Σ^1_1 is not closed by (classical) negation, it is clear that classical negation cannot be defined in IF.

Classical negation plays an important role in Hintikka's original programme. In [4], he claims that "virtually all off classical mathematics can in principle be done in extended IF first-order logic" (in a way that is ultimately "reducible" to plain IF logic). What he calls "(truth-functionally) extended IF logic" is the closure of the set of IF-sentences with operators $\neg$, $\wedge$ and $\vee$. Clearly, extended IF logic corresponds in expressive power to the boolean closure of Σ^1_1, which is known to be a proper fragment of Δ^1_2 [2,10].

In this paper we will consider the extension of IF logic where classical negation can be combined with IF-operators without restrictions (in contrast to Extended IF, where formulas with $\neg$ are obtained from boolean combination of IF sentences). One might suspect the resulting logic to be extremely expressive: freely combining classical negation with second order existential quantifiers leads to full second-order logic (SO). This suspicion is true for the logic with Henkin quantifiers when the set of available quantifiers is closed by a dualization operator [9]. Indeed, this logic has the same expressive power as SO. We will show that, in our context, this is not the case: IF with unrestricted classical negation is not more expressive than Δ^1_2.

We take as a basis Hodges' treatment of classical negation [6]. He defines an extension of IF, called *slash logic* (SL), in which independence restrictions can occur in any connective (and not only $\exists$ and $\vee$) and provide a compositional semantics for it. To support classical negation, the *flattening* operator $\downarrow$ is introduced. Intuitively, this operator satisfies:

$$\mathcal{A} \models^{-} {\downarrow}\varphi \text{ iff } \mathcal{A} \not\models^{+} \varphi \tag{4}$$

It is easy to see that classical negation can be expressed combining $\sim$ and $\downarrow$:

$$\mathcal{A} \models^{+} \neg\varphi \text{ iff } \mathcal{A} \models^{+} {\sim}{\downarrow}\varphi \text{ iff } \mathcal{A} \models^{-} {\downarrow}\varphi \text{ iff } \mathcal{A} \not\models^{+} \varphi \tag{5}$$

Let SL($\downarrow$) denote the extension of SL with the flattening operator. We will use the game-theoretical semantics introduced in [3] (equivalent to Hodges' compositional semantics), which we present in §2. From this we derive a prenex normal form result in §3 from which a Skolem form for SL($\downarrow$)-formulas is obtained in §4. By analyzing the syntactic fragment of SO in which the Skolem form falls, we ultimately arrive to the Δ^1_2 upper-bound.

2 Syntax and Semantics of SL(↓)

We assume a fixed first-order relational language $\mathcal{L}$, as well as a collection X of first-order variables, which we will denote x, y, z, perhaps with subindices. Formulas of SL(↓), in negation normal form, correspond to the following grammar:

$$\varphi ::= l(x_1, \ldots, x_k) \mid \exists x_{i|\rho}\varphi \mid \forall x_{i|\rho}\varphi \mid \downarrow\varphi \mid \uparrow\varphi \mid \varphi \vee_{|\rho} \varphi \mid \varphi \wedge_{|\rho} \varphi \qquad (6)$$

where ρ denotes a (possibly empty) finite set of variables and $l(x_1, \ldots, x_k)$ is any first-order literal (i.e., an atom or a negated atom). We will typically use $\exists x_i$, $\forall x_i$, $\vee$ and $\wedge$ instead of $\exists x_{i|\emptyset}$, $\forall x_{i|\emptyset}$, $\vee_{|\emptyset}$ and $\wedge_{|\emptyset}$. Since we are working in negation normal form, (game) negation $\sim$ will be a mapping on functions satisfying $\sim\forall x_{i|\rho}\varphi = \exists x_{i|\rho}\sim\varphi$; $\sim\downarrow\varphi = \uparrow\sim\varphi$, etc. Finally, $\neg\varphi$ will be short for $\sim\downarrow\varphi$. $\mathrm{Fv}(\varphi)$ denotes the set of free variables of φ (see [3] for a formal definition). A *sentence* is a formula with no free variables.

Remark 1. For the sake of simplicity we will impose a further restriction on formulas: there can be no nested bindings of the same variable (e.g., $\exists x \exists x \varphi$) nor a variable that occurs both free and bound in a formula (e.g., $x \vee \exists x \varphi$). This is called the *regular fragment* of SL(↓) and it has simpler formal semantics. The results in this paper apply to the whole language under the proviso that *history-preserving valuations* are used instead of standard ones (cf. [3] for details).

We interpret a SL(↓)-formulas using first-order models $\mathcal{A}$ with domain $|\mathcal{A}|$. We use sets of *finite* valuations to account for free variables; the domains of these valuations must be large enough to interpret them all (but they can be larger).

Definition 1. *Given φ and $\mathcal{A}$, we say that, V, a set of finite valuations over $\mathcal{A}$, is* suitable *for φ iff there is a finite set $D \supseteq \mathrm{Fv}(\varphi)$ such that $V \subseteq |\mathcal{A}|^D$.*

We define now the game $\mathsf{G}(\mathcal{A}, \varphi, V)$, where $\mathcal{A}$ is a model and V is a set of finite valuations over $\mathcal{A}$ suitable for φ. Furthermore, let $\tilde{D} \supseteq \mathrm{Fv}(\varphi)$ be such that $V \subseteq |\mathcal{A}|^{\tilde{D}}$. As is customary, this game is played between two opponents: Eloïse and Abélard (sometimes called *Verifier* and *Falsifier*). There is also a third agent, called Nature, which acts either as a generator of random choices or as a referee.

The board. Game $\mathsf{G}(\mathcal{A}, \varphi, V)$ is played over the syntactic tree of φ. There is, additionally, a placeholder for a set D and a valuation $v : D \to |A|$. Initially, D is set to $\tilde{D}$ and v is empty. In the syntactic tree of φ, all the $\exists$, $\vee$ and $\downarrow$-nodes of the tree *belong* to Eloïse; while the $\forall$, $\wedge$ and $\uparrow$-nodes belong to Abélard. Moreover, $\exists$, $\forall$, $\vee$ and $\wedge$-nodes will be (repeatedly) decorated with functions during the game; the first two admit any function $f : |\mathcal{A}|^D \to |\mathcal{A}|$; the last two, only functions $f : |\mathcal{A}|^D \to \{L, R\}$. Initially, nodes have no decoration.

The turns. At any point of the game, the remaining number of turns is bounded by the maximum number of nested occurrences of ↓-nodes and ↑-nodes in the game-board.

- *The opening turn.* The first turn is different from the rest. It is composed of two clearly distinguished phases. In the first phase, both players decorate all their nodes with suitable functions. The order in which they tag their nodes is not important as long as they do not get to see their opponent's choices in advance. For simplicity, we will assume they both play simultaneously. In the second phase, Nature picks a valuation from V and puts it in the placeholder v and finally *evaluates* the outcome of the turn, as described below.
- *The subsequent turns.* In all but the first turn, the formula tree is of the form $\downarrow\psi$ or $\uparrow\psi$ (see next). In these turns, both players get to redecorate their nodes, one after the other; Eloïse goes first when the formula tree is of the form $\downarrow\psi$ and Abélard does so on $\uparrow\psi$. Finally, Nature replaces the tree with ψ and proceeds to evaluate.

The recursive evaluation procedure used by Nature is the following:

R1 If the tree is of the form $\psi_1 \vee_{|y_1,\dots,y_k} \psi_2$ or $\psi_1 \wedge_{|y_1,\dots,y_k} \psi_2$, then its root must have been with a function $f : |\mathcal{A}|^D \to \{L, R\}$. Nature picks elements $a_1 \dots a_k$ from $|\mathcal{A}|$ and evaluates $f(v[y_1 \mapsto a_1, \dots, y_k \mapsto a_k])$ —by construction $\{y_1, \dots y_k\} \subseteq D$. That is, the values the player was not supposed to consider are randomly replaced prior to evaluating the function provided. The tree is then updated with ψ_1, if the result is L, and with ψ_2, otherwise. D and v remain unchanged and evaluation proceeds.

R2 If the tree is of the form $\exists x_{|y_1,\dots,y_k}\psi$ or $\forall x_{|y_1,\dots,y_k}\psi$, then it must have been decorated with a function $f : |\mathcal{A}|^D \to |\mathcal{A}|$. Again, Nature picks $a_1 \dots a_k$, evaluates $b := f(v[y_1 \mapsto a_1, \dots, y_k \mapsto a_k])$ and records this choice by replacing the placeholder with $D := D \cup \{x\}$ and $v := v \cup \{x \mapsto b\}$. Finally, the tree is updated with ψ and evaluation proceeds.

R3 If the tree is of the form $\downarrow\psi$ or $\uparrow\psi$, the evaluation –and, thus, the turn– ends.

R4 Finally, if the root of the tree is a literal $l(x_1, \dots, x_k)$, the game ends. Eloïse is declared the winner if $\mathcal{A} \models l(x_1, \dots, x_k)[v]$; otherwise, Abélard wins.

Nodes may get redecorated during the game but only by its owner, that is fixed. Hence it is equivalent to assume that players decorate only those nodes that are not under nested $\downarrow$ or $\uparrow$. This way, each node gets decorated only once.

Winning strategies. We will not go into a formal description of what a strategy for $\mathsf{G}(\mathcal{A}, \varphi, V)$ is. We simply take it to be a form of oracle that tells the player how to proceed in each turn. As usual, a strategy is said to be *winning* for a player if it guarantees that the he or she will win every match of the game, regardless the strategy of the opponent and the choices made by Nature.

We are now ready to give our game-semantic notion of *truth* and *falsity.*

Definition 2. *Let V be a set of finite valuations suitable for φ. We define:*

- $\mathcal{A} \models^+ \varphi[V]$ *iff Eloïse has a winning strategy for the game* $\mathsf{G}(\mathcal{A}, \varphi, V)$*;*
- $\mathcal{A} \models^- \varphi[V]$ *iff Abélard has a winning strategy for the game* $\mathsf{G}(\mathcal{A}, \varphi, V)$*.*

When $V = \{v\}$ we may alternatively write $\mathcal{A} \models^{+} \varphi[v]$ and $\mathcal{A} \models^{-} \varphi[v]$. Also, for a *sentence* φ we may write $\mathcal{A} \models^{+} \varphi$ and $\mathcal{A} \models^{-} \varphi$ meaning $\mathcal{A} \models^{+} \varphi[\varnothing]$ and $\mathcal{A} \models^{-} \varphi[\varnothing]$, respectively, where $\varnothing$ is the empty valuation (i.e. *dom* $\varnothing = \emptyset$). Unlike classical logics, from $\mathcal{A} \models^{+} \varphi[\varnothing]$ we cannot infer $\mathcal{A} \models^{+} \varphi[v]$ for every suitable v. This is due to *signaling*: the value of a variable a player is supposed not to know is available through the value of another one (cf. [7,8]).

Definition 3. *We write $\varphi_1 \equiv \varphi_2$ whenever, for every $\mathcal{A}$ and every set V suitable for φ_1 and φ_2, $\mathcal{A} \models^{+} \varphi_1[V]$ iff $\mathcal{A} \models^{+} \varphi_2[V]$ and $\mathcal{A} \models^{-} \varphi_1[V]$ iff $\mathcal{A} \models^{-} \varphi_2[V]$.*

When interested in winning strategies for, say, Eloïse in $\mathsf{G}(\mathcal{A}, \varphi, \{v\})$, it makes no difference whether in the opening turn both players play simultaneously or if Eloïse goes first. This is not true, though, in $\mathsf{G}(\mathcal{A}, \varphi, V)$ for a non-singleton V.

Proposition 1. *Given v, a finite valuation over $\mathcal{A}$ suitable for φ, we have that $\mathcal{A} \models^{+} \varphi[v]$ iff $\mathcal{A} \models^{+} \downarrow\varphi[v]$, and that $\mathcal{A} \models^{-} \varphi[v]$ iff $\mathcal{A} \models^{-} \uparrow\varphi[v]$.*

Remark 2. Whenever one is interested in whether $\mathcal{A} \models^{+} \varphi[V]$ holds or not, it is clearly equivalent (and sometimes convenient, too) to consider a simplified version of $\mathsf{G}(\mathcal{A}, \downarrow\varphi, V)$ in which Eloïse plays functions and Abélard plays *elements* (until the game reaches a $\uparrow$, where the situation gets reversed). This resembles the perfect-information game for IF given by Väänänen in [11].

Operator $\downarrow$ turns a formula that may lead to a non-determined game, into one that always leads to a determined one. This suggests the following notion.

Definition 4. *We say that φ is* determined *whenever, for every model $\mathcal{A}$, and every set V suitable for φ, $\mathcal{A} \not\models^{+} \varphi[V]$ iff $\mathcal{A} \models^{-} \varphi[V]$.*

Intuitively, determined formulas are those that have a well-defined truth-value on every structure. Plain first-order formulas (i.e., those with no independence restrictions) are determined, but one can give more general conditions.

Proposition 2. *The following hold:*

1. *Every literal is a determined formula.*
2. *$\downarrow\psi$ and $\uparrow\psi$ are determined formulas.*
3. *If φ and ψ are determined formulas, so are $\varphi \wedge_{|\emptyset} \psi$, $\varphi \vee_{|\emptyset} \psi$, $\exists x_{|\emptyset}\varphi$ an $\forall x_{|\emptyset}\varphi$.*

3 Normal Forms for SL($\downarrow$)

Normal forms in the context of SL were initially investigated in [1]. Later, Janssen [7] observed some anomalies which cast doubt on the correctness of these results. However, it was shown in [3] that only the formal apparatus employed in [1] was defective, and not the results per se.

In this section we revisit the prenex normal form results of [1] and extend them to account for $\downarrow$ and $\uparrow$. For this, bound variables will be tacitly renamed when necessary[1] and the following formula manipulation tools will be employed.

[1] While this assumption was considered problematic in the context of [1], it is safe here since we are using regular formulas. Moreover, this can also be assumed for arbitrary formulas under an adequate formalization (cf. Remark 1).

Definition 5. *Let $x_1 \dots x_n$ be variables not occurring in φ; we denote with $\varphi_{|x_1 \dots x_n}$ the formula obtained by adding $x_1 \dots x_n$ as restrictions to every quantifier, every conjunction and every disjunction in φ. Also, we write φ^c for the formula obtained by replacing all independence restrictions in φ by $\emptyset$.*

Notice that φ^c is essentially a FO formula. As is observed in [1], independence restrictions on boolean connectives can be removed by introducing additional quantifications. It is not hard to extend this result to SL(↓) (in what follows, we shall use, for emphasis, $\vee_{|\emptyset}$ instead of $\vee$, $\wedge_{|\emptyset}$ instead of $\wedge$, etc.).

Theorem 1. *For every formula φ, there exists a φ' such that $\varphi \equiv \varphi'$ and every disjunction (resp. conjunction) in φ' is of the form $\psi_1 \vee_{|\emptyset} \psi_2$ (resp. $\psi_1 \wedge_{|\emptyset} \psi_2$).*

Proof. When restricted to models with at least two elements, a simple inductive argument gives us the desired formula. The important step is that, given a formula $\psi := \psi_1 \vee_{|x_1 \dots x_k} \psi_2$ and given y_1, y_2 fresh in ψ, we can define

$$\psi^* := \exists {y_1}_{|x_1 \dots x_k} \exists {y_2}_{|y_1, x_1 \dots x_n} [(y_1 = y_2 \wedge {\psi_1}_{|y_1,y_2}) \vee (y_1 \neq y_2 \wedge {\psi_2}_{|y_1,y_2})] \tag{7}$$

On models with at least two elements, we have $\psi \equiv \psi^*$. By successively applying this truth-preserving transformation in a top-down manner, one can obtain, for any given φ, a formula φ^* that is equivalent on models with at least two elements.

Now, observe that on models with exactly one element, restrictions are meaningless. Therefore, for any given φ we can define the equivalent formula:

$$\varphi' := (\forall x \forall y [x = y] \wedge \varphi^c) \vee (\exists x \exists y [x \neq y] \wedge \varphi^*) \tag{8}$$

Formula (7) in the above proof was taken from [1], with the proviso that independences on y_1 and y_2 are added to ψ_1 and ψ_2 in order to avoid unwanted signaling [7,8,3]. This was most probably an involuntary omission in [1].

The prenex normal form result for SL in [1] uses a lemma that we will need.

Lemma 1 ([1]). *If x does not occur in ψ, then the following equivalences hold:*

1. $\exists x_{|\rho}[\varphi] \vee_{|\emptyset} \psi \equiv \exists x_{|\rho}[\varphi \vee_{|\emptyset} \psi_{|x}]$.
2. $\exists x_{|\rho}[\varphi] \wedge_{|\emptyset} \psi \equiv \exists x_{|\rho}[\varphi \wedge_{|\emptyset} \psi_{|x}]$.
3. $\forall x_{|\rho}[\varphi] \vee_{|\emptyset} \psi \equiv \forall x_{|\rho}[\varphi \vee_{|\emptyset} \psi_{|x}]$.
4. $\forall x_{|\rho}[\varphi] \wedge_{|\emptyset} \psi \equiv \forall x_{|\rho}[\varphi \wedge_{|\emptyset} \psi_{|x}]$.

For SL(↓), we need to show how to extract ↓ and ↑ from arbitrary formulas.

Lemma 2. *The following hold:*

1. *If ψ is determined, then $\downarrow\psi \equiv \uparrow\psi \equiv \psi$.*
2. $\downarrow(\varphi \wedge_{|\emptyset} \psi) \equiv \downarrow\varphi \wedge_{|\emptyset} \downarrow\psi$ *and* $\uparrow(\varphi \wedge_{|\emptyset} \psi) \equiv \uparrow\varphi \wedge_{|\emptyset} \uparrow\psi$.
3. $\downarrow(\varphi \vee_{|\emptyset} \psi) \equiv \downarrow\varphi \vee_{|\emptyset} \downarrow\psi$ *and* $\uparrow(\varphi \vee_{|\emptyset} \psi) \equiv \uparrow\varphi \vee_{|\emptyset} \uparrow\psi$.

Proof. We shall only discuss $\downarrow(\varphi \wedge_{|\emptyset} \psi) \equiv \downarrow\varphi \wedge_{|\emptyset} \downarrow\psi$; the remaining equivalences are similar. Suppose, then, that $\mathcal{A} \models^+ \downarrow(\varphi \wedge_{|\emptyset} \psi)[V]$; this means that Eloïse has a way of decorating both φ and ψ that guarantees she wins both games. Therefore, we have $\mathcal{A} \models^+ \downarrow\varphi[V]$ and $\mathcal{A} \models^+ \downarrow\psi[V]$ which implies $\mathcal{A} \models^+ (\downarrow\varphi \wedge_{|\emptyset} \downarrow\psi)[V]$. The right to left direction is analogous, and one establishes that $\mathcal{A} \models^+ \downarrow(\varphi \wedge_{|\emptyset} \psi)[V]$ iff $\mathcal{A} \models^+ (\downarrow\varphi \wedge \downarrow\psi)[V]$. Since $\downarrow(\varphi \wedge_{|\emptyset} \psi)$ and $(\downarrow\varphi \wedge_{|\emptyset} \downarrow\psi)$ are determined formulas (Proposition 2), this implies $\mathcal{A} \models^- \downarrow(\varphi \wedge_{|\emptyset} \psi)[V]$ iff $\mathcal{A} \models^- (\downarrow\varphi \wedge \downarrow\psi)[V]$.

Definition 6. *A SL(↓)-formula is said to be in prenex normal form if it is of the form* $Q_0^* \updownarrow_1 Q_1^* \updownarrow_2 \dots Q_{n-1}^* \updownarrow_{n-1} Q_n^* \varphi$ *with* $n \geq 0$*, where each* Q_i^* *is a (perhaps empty) chain of quantifiers,* $\updownarrow_i \in \{\downarrow, \uparrow\}$ *and* φ *contains only literals,* $\wedge_{|\emptyset}$ *and* $\vee_{|\emptyset}$*.*

Theorem 2. *For every* φ*, there exists a* φ^* *in prenex normal form with* $\varphi \equiv \varphi^*$*.*

Proof. By Theorem 1 we can obtain a φ' such that $\varphi' \equiv \varphi$ and no boolean connective in it contains independences. We proceed now by induction on φ'. If φ' is a literal, $\varphi^* = \varphi'$. If $\varphi' = \exists x_{|y_1 \dots y_k} \psi$, we have $\varphi^* = \exists x_{|y_1 \dots y_k} \psi^*$ and the cases for $\varphi' = \forall x_{|y_1 \dots y_k} \psi$, $\varphi' = \downarrow\psi$ and $\varphi' = \uparrow\psi$ are analogous. We analyze now the case for $\varphi' = \psi \vee \chi$; the one for $\varphi' = \psi \wedge \chi$ is symmetrical.

We need to show that there exists a $\varphi^* \equiv (\psi^* \vee \chi^*)$, in prenex normal form. We do it by induction on the sum of the lengths of the prefixes of ψ^* and χ^*. The base case is trivial; for the inductive case we show that one can always "extract" the outermost operator of either ψ^* or χ^*.

The first thing to note is that if $\psi^* = Qx_{|y_1 \dots y_k} \psi'$ ($Q \in \{\forall, \exists\}$), then using Lemma 1 (renaming variables, if necessary) we have $\varphi^* := Qx_{|y_1 \dots y_k} (\psi' \vee \chi^*)^*$ and the same applies to the case $\chi^* = Qx_{|y_1 \dots y_k} \chi'$. So suppose now that neither ψ^* nor χ^* has a quantifier as outermost operator. In that case, they start with one of ↓ or ↑, or they contain only $\wedge_{|\emptyset}$, $\vee_{|\emptyset}$ and literals. In either case, they are both determined and at least one of them starts with ↓ or ↑ (or we would be in the base case). If we assume that $\psi^* = \downarrow\psi'$, using Proposition 2 repeatedly, we have $(\downarrow\psi' \vee \chi^*) \equiv (\downarrow\psi' \vee \downarrow\chi^*) \equiv \downarrow(\psi' \vee \downarrow\chi^*) \equiv \downarrow(\psi' \vee \chi^*)$, and we can apply the inductive hypothesis. The remaining cases are analogous.

4 Skolem Forms, Dependencies and Δ_2^1-Expressivity

We finally turn our attention to the expressive power of SL(↓). We shall see that every SL(↓)-formula is equivalent to both Σ_2^1 and Π_2^1-formulas, and therefore, is included in Δ_2^1. We reserve letters f, g and h (probably with subindices) to denote second-order functional variables; arities will be left implicit. We identify first-order variables with 0-ary second-order variables; letters x, y and z (with subindices) are to be interpreted always as 0-ary functions (f, g, etc. could be 0-ary too, unless stated). We also assume that functional symbols occur always fully-saturated (e.g., $f = g$ is not a valid formula, but $\forall x[f(x) = g(x)]$ is).

4.1 Skolem Form of SL(↓) Sentences

Based on the game-theoretical semantics of §2 we show that every *sentence* of SL(↓) can be turned into an equivalent SO formula. We will motivate this sort of Skolem form with a short example. For this, let φ be quantifier-free, with variables among $\{x_1, x_2, x_3, y_1, y_2, y_3\}$ and consider:

$$\chi_2 := \downarrow \forall y_1 \forall y_2 \exists x_{1|y_2} \uparrow \exists x_{2|y_2} \exists x_3 \forall y_{3|x_3} [\varphi] \tag{9}$$

Assume that $\mathcal{A} \models^+ \chi_2$, i.e., that Eloïse has a winning strategy for $\mathsf{G}(\mathcal{A}, \chi_2, \{\varnothing\})$. Using the simplification of Remark 2 this is the case if and only if $\mathcal{A} \models \chi_2'$, where

$$\chi_2' := \exists f \forall y_1 \forall y_2 \forall z_1 \forall g \exists x_2 \exists x_3 \exists z_2 [\varphi \sigma_1] \tag{10}$$

and $\sigma_1 = \{x_1 \mapsto f(y_1, z_1), y_3 \mapsto g(y_1, y_2, f(y_1, z_1), x_2, z_2)\}$ is a substitution of variables by terms[2]. Notice that z_1 and z_2 represent the random choices made by Nature during the evaluation phases; e.g., $f(y_1, z_1)$ expresses that Nature replaced the value of y_2 by a randomly picked z_1 when evaluating x_1. Since z_1 and z_2 do not occur in φ and y_1 and y_2 occur universally quantified, just as g, we have that χ_2' is equivalent to χ_2'', where

$$\chi_2'' := \exists f \forall y_1 \forall y_2 \forall g \exists x_2 \exists x_3 [\varphi \sigma_2] \tag{11}$$

and $\sigma_2 = \{x_1 \mapsto f(y_1), y_3 \mapsto g(x_2, f(y_1))\}$. Of course, one could simplify further and replace $g(x_2, f(y_2))$ by $g(x_2)$, but this will be discussed in more detail in §4.2.

In order to formalize this transformation, we will use some conventions. First, λ denotes an empty sequence (of quantifiers, of variables, etc.). When describing SL(↓) prefixes we shall use patterns such as $\downarrow \forall y_{1|\tau_1} \exists x_{1|\rho_1} \dots \forall y_{k|\tau_k} \exists x_{k|\rho_k} \updownarrow Q$; it must be understood that not necessarily all the x_i and y_i are present in the prefix, and that either $\updownarrow Q = \lambda$ or else $\updownarrow \in \{\downarrow, \uparrow\}$ and Q is a (possibly empty) SL(↓)-prefix.

Definition 7. *Given $Q\psi$ in prenex normal form (ψ quantifier-free), $T(Q\psi)$ is the SO-formula $\pi(\downarrow Q)[\psi\sigma]$, where $\sigma = \tau^\lambda_\lambda(\{\}, \downarrow Q)$, $\pi(\lambda) = \lambda$, $\tau^\alpha_\beta(\sigma, \lambda) = \sigma$ and*

$$\begin{aligned}
\pi(\downarrow \forall y_{1|\tau_1} \exists x_{1|\rho_1} \dots \forall y_{k|\tau_k} \exists x_{k|\rho_k} \updownarrow Q) &= \exists f_1 \dots \exists f_k \forall y_1 \dots \forall y_k \pi(\updownarrow Q) \\
\pi(\uparrow \exists x_{1|\rho_1} \forall y_{1|\tau_1} \dots \exists x_{k|\rho_k} \forall y_{k|\tau_k} \updownarrow Q) &= \forall g_1 \dots \forall g_k \exists x_1 \dots \exists x_k \pi(\updownarrow Q) \\
\tau^\alpha_\beta(\sigma, \downarrow \forall y_{1|\tau_1} \exists x_{1|\rho_1} \dots \forall y_{k|\tau_k} \exists x_{k|\rho_k} \updownarrow Q) &= \tau^{\alpha'}_{\beta'}(\sigma \cup \{x_i \mapsto f_{x_i}(\alpha' \setminus \rho_i)\sigma\}, \updownarrow Q) \\
\tau^\alpha_\beta(\sigma, \uparrow \exists x_{1|\rho_1} \forall y_{1|\tau_1} \dots \exists x_{k|\rho_k} \forall y_{k|\tau_k} \updownarrow Q) &= \tau^{\alpha'}_{\beta'}(\sigma \cup \{y_i \mapsto g_{y_i}(\beta' \setminus \tau_i)\sigma\}, \updownarrow Q)
\end{aligned}$$

Here, we assumed $\alpha' := \alpha, y_1 \dots y_k$ and $\beta' := \beta, x_1 \dots x_k$.

The reader should verify that, modulo variable names, $T(\chi_2) = \chi_2''$. In particular, substitution application in $f_{x_i}(\alpha' \setminus \rho_i)\sigma$ and $g_{y_i}(\beta' \setminus \rho_i)\sigma$ account for the introduction of nested terms like $g(x_2, f(y_1))$ in (11).

Lemma 3. *For every sentence φ in prenex normal form and every model $\mathcal{A}$, $\mathcal{A} \models^+ \varphi$ iff $\mathcal{A} \models T(\varphi)$.*

Proof. First, observe that $\mathcal{A} \models^+ \varphi$ iff $\mathcal{A} \models^+ \downarrow\varphi$ (Proposition 1). One then can show that, for every ψ in prenex normal form (perhaps with free variables) and every suitable v, $\mathcal{A} \models^+ \downarrow\psi[v]$ iff $\mathcal{A} \models T(\downarrow\psi)[v]$ by induction on the number of turns in $\mathsf{G}(\mathcal{A}, \downarrow\psi, \{v\})$ (i.e., in the number of ↓ and ↑ occurring in $\downarrow\psi$).

It follows directly from Definition 7 that $T(\varphi)$ is a SO formula in prenex normal form (i.e., all quantifiers, either first or second-order, are at the beginning).

[2] As is customary, we use postfix notation for substitution application.

4.2 First-Order Dependencies and the Δ^1_2-Class

If φ, in prenex normal form, has d occurrences of $\downarrow$ or $\uparrow$, then $T(\varphi)$ is a formula in Σ^1_{2d+1}. But we will see that by repeatedly applying a truth-preserving quantifier-moving transformation, one can turn any such $T(\varphi)$ into an equivalent formula in Σ^1_2. Since SL($\downarrow$) is closed by negation, this gives us the desired Δ^1_2-bound. We begin with a motivational a example. Let φ be quantifier-free and consider:

$$\chi_3 := \downarrow\forall y_1 \exists x_1 \uparrow \exists x_2 \forall y_{2|x_1} \downarrow \forall y_3 \exists x_{3|y_1,y_2} \uparrow \exists x_4 \forall y_{4|x_1\ldots x_3} \varphi \tag{12}$$

$$\chi'_3 := T(\chi_3) = \exists f_1 \forall y_1 \forall g_1 \exists x_2 \exists f_2 \forall y_3 \forall g_2 \exists x_4 [\varphi\sigma_3] \tag{13}$$

where $\sigma_3 = \{x_1 \mapsto f_1(y_1), y_2 \mapsto g_1(x_2), x_3 \mapsto f_2(y_3), y_4 \mapsto g_2(x_4)\}$. The key intuition to show that χ'_3 is a Σ^1_2-formula is that the choice for f_2 does not actually depend on the chosen value for g_1 but only on that of $g_1(x_2)$, which is the only occurrence of g_1 in $\varphi\sigma_3$. Thus, χ'_3 is equivalent to:

$$\chi_4 := \exists f_1 \exists \tilde{f}_2 \forall y_1 \forall g_1 \forall y_3 \exists x_2 \forall g_2 \exists x_4 [\varphi\sigma_4] \tag{14}$$

where $\sigma_4\{x_1 \mapsto f_1(y_1), y_2 \mapsto g_1(x_2), x_3 \mapsto \tilde{f}_2(y_1, g_1(x_2), y_3), y_4 \mapsto g_2(x_4)\}$. Finally, one can "exchange" g_2 and x_2 in an analogous way, to obtain a Σ^1_2-formula.

We identify next the fragment of SO where this "weak-dependency" between functions occurs and show that, in it, the procedure sketched above can always be performed. In what follows we assume, without loss of generality, that if a variable occurs free in a SO-formula, it does not also appear bound.

Definition 8. *We say that a functional symbol f is* strongly free *in a SO-formula φ whenever f is free in φ and, if $f(\ldots g(\ldots)\ldots)$ occurs in φ, g is strongly free in φ too.*

As an example, g is strongly free in $\exists x[f(x) = g(y, g(z, y))]$, while f is not (for x is not, either). Every free first-order variable is also strongly free.

Lemma 4. *Let φ be a SO-formula and let $g_1 \ldots g_k$ be strongly free in φ. Moreover, let v_1 and v_2 be interpretations of functional variables in $\mathcal{A}$ such that (i) $v_1(f) = v_2(f)$ for every $f \notin \{g_1, \ldots, g_k\}$, and (ii) for every $g_i(t_1, \ldots, t_m)$ occurring in φ, $v_1(g_i(t_1, \ldots, t_m)) = v_2(g_i(t_1, \ldots, t_m))$. Then, $\mathcal{A} \models \varphi[v_1]$ iff $\mathcal{A} \models \varphi[v_2]$.*

Proof. Follows by a straightforward induction on φ. In the base case, (i) and (ii) guarantee the equivalence; for the case $\varphi = \exists f \psi$, removing the existential preserves the hypothesis of the lemma (needed for the inductive hypothesis).

It is well-known that $\forall x_1 \ldots x_n \exists f \varphi$ is equivalent to $\exists \tilde{f} \forall x_1 \ldots x_n \tilde{\varphi}$, where $\tilde{\varphi}$ is obtained by replacing every occurrence of a term of the form $f(t_1, \ldots, t_k)$ in φ by $\tilde{f}(t_1, \ldots, t_k, x_1, \ldots, x_n)$. The following result is a generalization of this idea, with strongly free second-order variables used instead of first-order ones.

Theorem 3. *Let $g_1 \ldots g_n$ be strongly free in φ and let h, free in φ, be such that for no i, $g_i(\ldots h(\ldots)\ldots)$ occurs in φ. Then, for every $f_1 \ldots f_m$ free in φ, there exists $\tilde{\varphi}$ such that $g_1 \ldots g_n$ are strongly free in $\tilde{\varphi}$; $f_1 \ldots f_m$ are free in $\tilde{\varphi}$ and $\forall g_1 \ldots \forall g_n \exists h \exists f_1 \ldots \exists f_k \varphi \equiv \exists \tilde{h} \forall g_1 \ldots \forall g_n \exists f_1 \ldots \exists f_k \tilde{\varphi}$.*

Proof. One can obtain $\tilde{\varphi}$ by replacing every occurrence of $h(t_1, \ldots, t_k)$ in φ by $\tilde{h}(t_1, \ldots, t_k, s_1, \ldots, s_l)$, where $s_1 \ldots s_l$ are the terms of the form $g_i(\ldots)$ occurring in φ. Since $g_i(\ldots h(\ldots) \ldots)$ does not occur in φ, no occurrence of h is left in $\tilde{\varphi}$.

In a way, what Theorem 3 says is that a strongly free second-order variable corresponds, in terms of *information*, to a finite number of first-order terms. This motivates the following definition.

Definition 9. *We say that a SO-formula in prenex normal form has* first-order dependencies *if in every subformula of the form* $\forall g_1 \ldots \forall g_n \exists f_1 \ldots \exists f_m \varphi$ *(with* $\varphi \neq \exists h \psi$*) or* $\exists g_1 \ldots \exists g_n \forall f_1 \ldots \forall f_m \varphi$ *(*$\varphi \neq \forall h \psi$*),* $g_1 \ldots g_n$ *are strongly free in* φ*.*

Theorem 4. *The class of SO-formulas with first-order dependencies is in* Δ^1_2*.*

Proof. We shall only see that if φ has first-order dependencies then φ is in Σ^1_2. This suffices since $\varphi \equiv \neg\neg\varphi$ and $\neg\varphi$ has first-order dependencies (and is therefore in Σ^1_2). To see that any such φ can be turned to Σ^1_2-form, we will show how quantifiers can be reordered, one at a time, in a top-down manner.

Suppose, then, that φ is not in Σ^1_2-form; and let us consider the leftmost quantifier that is misplaced. There are two cases we need to consider. First, suppose $\varphi = \exists h_1 \ldots \exists h_l \forall g_1 \ldots \forall g_n \exists x_1 \ldots \exists x_k \exists h \exists f_1 \ldots \exists f_m \psi$ ($k, l, m \geq 0$, $n \geq 1$) for an h of *non-zero arity.* We can assume without loss of generality that $g_i(\ldots h(\ldots) \ldots)$ does not occur in ψ, e.g., if $g_i(h(x_j))$ occurs in ψ, then φ is equivalent to $\exists h_1 \ldots \exists h_l \forall g_1 \ldots \forall g_n \exists x_1 \ldots \exists x_k \exists h \exists z \exists f_1 \ldots \exists f_m [z = h(x_j) \wedge \chi]$ where χ is like ψ with $h(x_j)$ replaced by z. By Theorem 3 we can relocate the misplaced $\exists h$ and obtain the equivalent formula $\exists h_1 \ldots \exists h_l \exists \tilde{h} \forall g_1 \ldots \forall g_n \exists x_1 \ldots \exists x_k \exists f_1 \ldots \exists f_m \tilde{\psi}$.

The second case is when every variable in the first block of misplaced quantifiers is first-order. Here we cannot always circumvent the condition "$g_i(\ldots x_j \ldots)$ does not occur in ψ" so we must handle it in a different way. Therefore, suppose $\varphi = \exists h_1 \ldots \exists h_l \forall g_1 \ldots \forall g_n \exists x_1 \ldots \exists x_k \forall f_1 \ldots \forall f_m \psi$, ($l \geq 0, k, m, n \geq 1$) where ψ does not start with $\forall$. Using Theorem 3 ($x_1 \ldots x_k$ are strongly free in ψ), we obtain the equivalent $\exists h_1 \ldots \exists h_l \forall g_1 \ldots \forall g_n \forall \tilde{f_1} \ldots \forall \tilde{f_m} \exists x \ldots \exists x_k \tilde{\psi}$. It is important to notice that $\forall g_1 \ldots \forall g_n \forall \tilde{f_1} \ldots \forall \tilde{f_m} \exists x \ldots \exists x_k \tilde{\psi}$ has first-order dependencies.

Corollary 1. $T(\varphi)$ *has first-order dependencies, for every SL(*$\downarrow$*)-formula* φ *in prenex normal form. Therefore, SL(*$\downarrow$*) is not more expressive than* Δ^1_2*.*

5 Conclusions

We have given an upper bound for the expressive power of SL($\downarrow$) and, therefore, for IF with full-fledged classical negation. This logic contains Extended-IF, which is known to correspond to a proper fragment of Δ^1_2; hence, our Δ^1_2-upper bound for SL($\downarrow$) is quite tight. This result is mounted on three additional results: *i)* a game-theoretical semantics for SL($\downarrow$), *ii)* a prenex normal form result for SL($\downarrow$) and, *iii)* a characterization of a new (to the best of our knowledge) syntactic fragment of SO (formulas with first-order dependencies) which we have shown to be contained in Δ^1_2. The last two are given in this article (for the first one, cf. [3]); the prenex normal form uses ideas from [1]. The question of which of the containments are strict remains open.

References

1. Caicedo, X., Krynicki, M.: Quantifiers for reasoning with imperfect information and Σ_1^1-logic. Contemporary Mathematics 235, 17–31 (1999)
2. Enderton, H.: Finite partially ordered quantifiers. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 16, 393–397 (1970)
3. Figueira, S., Gorín, D., Grimson, R.: On the formal semantics of IF-like logics. Journal of Computer and System Sciences 76(5), 333–346 (2009)
4. Hintikka, J.: The Principles of Mathematics Revisited. Cambridge University Press, Cambridge (1996)
5. Hintikka, J., Sandu, G.: Game-theoretical semantics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language. The MIT press, Cambridge (1997)
6. Hodges, W.: Compositional semantics for a language of imperfect information. Logic Journal of the IGPL 5(4) (1997)
7. Janssen, T.M.V.: Independent choices and the interpretation of IF logic. Journal of Logic, Language and Information 11(3), 367–387 (2002)
8. Janssen, T.M.V., Dechesne, F.: Signalling in IF games: a tricky business. In: The Age of Alternative Logics, pp. 221–241. Springer, Heidelberg (2006)
9. Kołodziejczyk, L.A.: The expressive power of Henkin quantifiers with dualization. Master's thesis, Institute of Philosophy, Warsaw University, Poland (2002)
10. Mostowski, M.: Arithmetic with the Henkin quantifier and its generalizations. In: Gaillard, F., Richard, D. (eds.) Séminaire du Laboratoire Logique, Algorithmique et Informatique Clermontoise, vol. 2, pp. 1–25 (1991)
11. Väänänen, J.A.: On the semantics of informational independence. Logic Journal of the IGPL 10(3), 339–352 (2002)

Concurrent Logic Games on Partial Orders

Julian Gutierrez

LFCS, School of Informatics, University of Edinburgh

Abstract. Most games for analysing concurrent systems are played on interleaving models, such as graphs or infinite trees. However, several concurrent systems have partial order models rather than interleaving ones. As a consequence, a potentially algorithmically undesirable translation from a partial order setting to an interleaving one is required before analysing them with traditional techniques. In order to address this problem, this paper studies a game played directly on partial orders and describes some of its algorithmic applications. The game provides a unified approach to system and property verification which applies to different decision problems and models of concurrency. Since this framework uses partial orders to give a uniform representation of concurrent systems, logical specifications, and problem descriptions, it is particularly suitable for reasoning about concurrent systems with partial order semantics, such as Petri nets or event structures. Two applications can be cast within this unified approach: bisimulation and model-checking.

1 Introduction

Games form a successful approach to giving semantics to logics and programming languages (semantic games) and to program verification (verification games). Good surveys of some of the most important game-based decision procedures and tools for property and systems verification can be found in [8,16,17], and in the references therein. These 'logic games' [5] usually are *sequential* and played on graphs or infinite trees. They offer an elegant approach to studying different properties of sequential processes and of concurrent systems with *interleaving* semantics, e.g., by using Kripke models or (labelled) transition graphs.

However, when dealing with concurrent systems with *partial order* semantics [15], such as Petri nets or event structures (which are semantically richer and more complex), the game-based techniques previously mentioned cannot be directly applied because the explicit notion of independence or *concurrency* in the partial order models is not considered. As a result, one has to construct the graph structures associated with those partial order models—a translation that one would like to avoid since, in many cases, it is algorithmically undesirable.

The reasons to wish to stay in a purely partial order setting are well-known by the concurrency theory community. For instance, partial order models of concurrency can be exponentially smaller than their interleaving counterparts; moreover they are amenable to partial order reductions [10] and are the natural input of the unfolding methods [7] for software and hardware verification—which work very well in practice whenever the systems have high degrees of parallelism.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 146–160, 2011.

Then, it is desirable, for several algorithmic reasons, to have a game which can be played directly on the partial order representations of concurrent systems. The main problem is that games played directly on 'noninterleaving structures' (which include Petri nets and event structures) are not known to be *determined* in the general case since they may well turn out to be of *imperfect* information, mainly, due to the information about *locality* and *independence* in such models.

In this paper we study a class of games played on noninterleaving structures (posets in our case) which is *sound* and *complete*, and therefore determined, without using stochastic strategies as traditional approaches to concurrent games [3,4]. Our framework builds upon two ideas: firstly, the use of posets to give a *uniform* representation of concurrent systems with partial order semantics, logical specifications, and problem descriptions; and secondly, the restriction to games with a *semantic* condition that reduces reasoning on different models and decision problems to the analysis of simpler *local* correctness conditions.

The solution is realised by a new 'concurrent logic game' (CLG) which is shown to be determined—even though it is, locally, of imperfect information. The two players of the game are allowed to make *asynchronous* and *independent* local moves in the board where the game is played. Moreover, the elements of the game are all formalised in order-theoretic terms; as a result, this new model builds a bridge between some concepts in order theory and the more operational world of games. To the best of our knowledge, such an order-theoretic characterisation has not been previously investigated for verification games.

Then our main contribution is the formalization of a concurrent logic game model that generalises the results in [16] to a partial order setting, that is, the games by Stirling for bisimulation and model-checking on interleaving structures (and hence also related tableau-based techniques). The CLG model is inspired by a concurrent semantic game model (for a fragment of Linear Logic [9]) studied by Abramsky and Melliès [2]. However, the mathematics of the original game have been drastically reformulated in the quest towards the answer to algorithmic questions, and only a few technical features were kept.

Acknowledgements. I thank J. Bradfield, L. Ong, I. Stark, C. Stirling, and G. Winskel for discussions and feedback on this and related manuscripts as well as the reviewers for their comments. An extended version, which includes proofs and examples that due to space could not be included here, is Chapter 5 of [12].

2 The Concurrent Game Model

We consider (infinite) logic games played by two players, Eve (∃) and Adam (∀), whose interaction can be used to represent the flow of information when analysing concurrent and distributed systems. The main idea is that by enriching a logic game with the explicit information about local and independent behaviour that comes with any partial order model, the traditional, sequential setting for logic games (usually played on interleaving structures) can be turned into a concurrent one on a partial order. This section studies a 'concurrent logic game' (CLG) played on a poset structure; a simple example is given in the appendix.

Preliminaries and Notations on Partial Orders. A $\bot_{\mathcal{A}}$-*bounded* poset $\mathfrak{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ is a partially ordered set with a *bottom* element $\bot_{\mathcal{A}}$ such that for all $a \in \mathcal{A}$ we have that $\bot_{\mathcal{A}} \leq_{\mathcal{A}} a$. For any $a \in \mathcal{A}$, a *successor* of a is an element a' such that $a <_{\mathcal{A}} a'$ and for all b if $a \leq_{\mathcal{A}} b$ and $b \leq_{\mathcal{A}} a'$ then either $a = b$ or $b = a'$. Write $a \rightarrow a'$ iff a' is a successor of a and call a a *terminal* element iff $a \not\rightarrow$. Given a, a (principal) *ideal* $\downarrow a$ is the downward-closed set $\{b \in \mathcal{A} \mid b \leq_{\mathcal{A}} a\}$; dually, a (principal) *filter* $\uparrow a$ of $\mathcal{A}$ is the upward-closed $\{b \in \mathcal{A} \mid a \leq_{\mathcal{A}} b\}$. Also, for any set $A \subseteq \mathcal{A}$, write $\downarrow A$ for the set $\bigcup_{a \in A}\{b \mid b \in \downarrow a\}$, and likewise, $\uparrow A$ for $\bigcup_{a \in A}\{b \mid b \in \uparrow a\}$; call $\downarrow A$ a *lower* subset and $\uparrow A$ an *upper* subset. We write $\downarrow a$ for the induced poset $(\downarrow a, \leq_{\mathcal{A}})$, and similarly for $\uparrow a$, $\downarrow A$, and $\uparrow A$. Clearly the posets $\downarrow a$ and $\uparrow a$ are $\bot$-bounded if $\mathfrak{A}$ is $\bot$-bounded, since $\bot_{\downarrow a} = \bot_{\mathcal{A}}$ in the former case and $\bot_{\uparrow a} = a$ in the latter. Finally, a function $f : \mathcal{A} \rightarrow \mathcal{A}$ is a *closure operator* iff it is extensive, monotonic, and idempotent, i.e., if satisfies that for all $a, a' \in \mathcal{A}$: $a \leq_{\mathcal{A}} f(a)$; $a \leq_{\mathcal{A}} a'$ implies $f(a) \leq_{\mathcal{A}} f(a')$; and $f(a) = f(f(a))$.

Game Boards. A board in a CLG model is a $\bot$-bounded poset $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ which is well-founded. A lower (resp. an upper) sub-board $\mathfrak{B}$ of $\mathfrak{D}$ is a poset $(\mathcal{B}, \leq_{\mathcal{D}})$ such that $\mathcal{B}$ is a lower (resp. an upper) subset of $\mathcal{D}$. Then, a lower sub-board is a finite or infinite $\bot$-bounded poset, whereas an upper sub-board is a union of possibly infinitely large $\bot$-bounded posets. In particular, since $\mathfrak{D}$ is well-founded, then all lower sub-boards are also well-founded. Moreover, a *global* position in $\mathfrak{D}$ is an anti-chain $D \subseteq \mathcal{D}$; the *initial* global position of $\mathfrak{D}$ is $\{\bot\}$. Finally, given a global position D in $\mathfrak{D}$, call any $d \in D$ a *local* position.

Notation 1. *Given any $d \in \mathcal{D}$, write $d^{\leftarrow}$ for the set of local positions $\{e \mid e \rightarrow d\}$ and $d^{\rightarrow}$ for the set $\{d' \mid d \rightarrow d'\}$. The sets $d^{\leftarrow}$ and $d^{\rightarrow}$ are, respectively, the 'preset' and 'postset' of local positions of d. Also, let $\mathtt{SP}(d)$ be the predicate that holds iff $\mid d^{\leftarrow} \mid > 1$, and call d a 'synchronization point' in such a case.*

Now, let $\nabla : \mathcal{D} \rightarrow \Upsilon$ be a partial function that assigns players in $\Upsilon = \{\exists, \forall\}$ to local positions. More precisely, ∇ is a total function on the set $\mathcal{B} \subseteq \mathcal{D}$ of elements that are not synchronization points—i.e., $\mathcal{B} = \{d \in \mathcal{D} \mid \neg\mathtt{SP}(d)\}$; call the pair $(\mathfrak{D}, \nabla)$ a *polarised* board. In the following we only consider polarised boards with the following property (called 'dsync'), which ensures that the behaviour at synchronization points is deterministic: $\mathtt{SP}(d) \Rightarrow |d^{\rightarrow}| = 1$ and $\forall e \in d^{\leftarrow}.|e^{\rightarrow}| = 1$.

This property, i.e., *dsync*, induces a correctness condition when playing the game. It ensures: firstly, that there are no choices to make in synchronization points (as they are not assigned by ∇ to any player); and secondly, that as a synchronization point does not share its preset with any other local position, then local behaviour in the game, which is formally defined later, is truly independent.

Strategies. In a CLG a strategy can be local or global. A *local* strategy $\lambda : \mathcal{D} \rightarrow \mathcal{D}$ is a closure operator partially defined on a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$. Being partially defined on $\mathfrak{D}$ means that the properties of closure operators are restricted to those elements where the closure operator is defined. In particular, the relation $a \leq_{\mathcal{D}} a' \Rightarrow \lambda(a) \leq_{\mathcal{D}} \lambda(a')$ holds iff λ is defined in a and a'. Let $\mathtt{dfn}(\lambda, d)$ be the predicate that holds iff $\lambda(d)$ is defined or evaluates to false otherwise. The predicate $\mathtt{dfn}$ can be defined from a board, for any local strategy, by means of three rules that realise local strategies $\lambda_{\forall}$ for Adam and $\lambda_{\exists}$ for Eve.

Definition 1 (Local Strategies). *Given a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, a 'local strategy' $\lambda_{\forall}$ for Adam (resp. $\lambda_{\exists}$ for Eve) is a closure operator defined only in those elements of $\mathcal{D}$ given by the following rules:*

1. *The local strategy $\lambda_{\forall}$ (resp. $\lambda_{\exists}$) is defined in the bottom element $\bot_{\mathcal{D}}$.*
2. *If $\lambda_{\forall}$ (resp. $\lambda_{\exists}$) is defined in $d \in \mathcal{D}$, and either $\nabla(d) = \exists$ (resp. $\nabla(d) = \forall$) or $\mathtt{SP}(d)$ or $d\not\rightarrow$ or $d \rightarrow e \wedge \mathtt{SP}(e)$ holds, then for all $d' \in d^{\rightarrow}$ we have that $\lambda_{\forall}$ (resp. $\lambda_{\exists}$) is defined in d' as well.*
3. *If $\lambda_{\forall}$ (resp. $\lambda_{\exists}$) is defined in $d \in \mathcal{D}$, and both $\nabla(d) = \forall$ (resp. $\nabla(d) = \exists$) and $|d^{\rightarrow}| \geq 1$ hold, then there exists a $d' \in d^{\rightarrow}$ in which $\lambda_{\forall}$ (resp. $\lambda_{\exists}$) is defined.*

Let $\mathtt{dfn}(\lambda_{\forall}, d)$ be the predicate that holds iff $\lambda_{\forall}(d)$ is defined, and likewise for $\lambda_{\exists}$. Moreover, the closed elements, i.e., the fixpoints, of $\lambda_{\forall}$ and $\lambda_{\exists}$ are as follows:

$$\lambda_{\forall}(d) = d \textit{ iff } \nabla(d) = \exists, \textit{ or } \mathtt{SP}(d), \textit{ or } d\not\rightarrow, \textit{ or } d \rightarrow e \wedge \mathtt{SP}(e)$$
$$\lambda_{\exists}(d) = d \textit{ iff } \nabla(d) = \forall, \textit{ or } \mathtt{SP}(d), \textit{ or } d\not\rightarrow, \textit{ or } d \rightarrow e \wedge \mathtt{SP}(e)$$

provided that the predicates $\mathtt{dfn}(\lambda_{\forall}, d)$ and $\mathtt{dfn}(\lambda_{\exists}, d)$ hold. Moreover, let $\lambda^{1}_{\forall}$ and $\lambda^{1}_{\exists}$ be the 'identity local strategies' of Adam and Eve, respectively, which are defined everywhere in $\mathfrak{D}$; thus, formally: $\lambda^{1}_{\forall}(d) = \lambda^{1}_{\exists}(d) = d$, for all $d \in \mathcal{D}$.

Let $\Lambda_{\mathfrak{D}}$ be the set of local strategies on $\mathfrak{D}$, which can be split in two subsets, i.e., $\Lambda_{\mathfrak{D}} = \Lambda^{\exists}_{\mathfrak{D}} \uplus \Lambda^{\forall}_{\mathfrak{D}}$, for Eve and Adam. Informally, Definition 1 says that a local strategy must be able (item 2) to reply to all 'counter-strategies' defined in the same local position, and (item 3) to choose a next local position whenever used. Moreover, item 3 of Definition 1 implies that in order for Eve and Adam to play concurrently, they have to follow a set of local strategies rather than only one.

Definition 1 also characterises the fixpoints of local strategies. Note that a *fixpoint* of a local strategy is a position in the board where a player cannot make a choice, either because it is the other player's turn (e.g., $\nabla(d) = \exists$ for $\lambda_{\forall}(d)$), or a synchronization must be performed ($\mathtt{SP}(d)$ or $d \rightarrow e \wedge \mathtt{SP}(e)$), or a terminal element is reached ($d\not\rightarrow$), and hence, there are no next local positions to play.

Remark 1. The intuitions as to why a closure operator captures the behaviour in a CLG follow [2]. As boards are *acyclic* ordered structures, there is no reason to move to a previous position and hence strategies should be extensive. They should also be monotonic in order to preserve the *causality* of moves in the game and idempotent to avoid *unnecessary* alternations between sequential steps.

When playing, Eve and Adam will use a set of local strategies $\Lambda^{\exists}_{Ↄ} \subseteq \Lambda^{\exists}_{\mathfrak{D}}$ and $\Lambda^{\forall}_{Ↄ} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$, whose elements (i.e., local strategies) are indexed by the elements i and j of two sets $\mathbb{K}_{\exists} = \{1, ..., |\Lambda^{\exists}_{\mathfrak{D}}|\}$ and $\mathbb{K}_{\forall} = \{1, ..., |\Lambda^{\forall}_{\mathfrak{D}}|\}$; by definition, the identity local strategies are indexed with $i = 1$ and $j = 1$. Moreover, at the beginning of the game Eve and Adam choose (independently and at the same time) two sets of indices $\mathrm{K}_{\exists} \subseteq \mathbb{K}_{\exists}$ and $\mathrm{K}_{\forall} \subseteq \mathbb{K}_{\forall}$, and consequently the two sets of local strategies $\Lambda^{\exists}_{Ↄ} \subseteq \Lambda^{\exists}_{\mathfrak{D}}$ and $\Lambda^{\forall}_{Ↄ} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$ they will use to play. This means that both $i \in \mathrm{K}_{\exists}$ iff $\lambda^{i}_{\exists} \in \Lambda^{\exists}_{Ↄ}$ and $j \in \mathrm{K}_{\forall}$ iff $\lambda^{j}_{\forall} \in \Lambda^{\forall}_{Ↄ}$; by definition, $\lambda^{1}_{\exists}$ and $\lambda^{1}_{\forall}$ are always included in $\Lambda^{\exists}_{Ↄ}$ and $\Lambda^{\forall}_{Ↄ}$. Based on this selection of local strategies one can define the sets of global strategies, reachable positions, and moves in the game.

Global strategies are interpreted in a poset of anti-chains since, by definition, a global position is an anti-chain of $\mathfrak{D}$. We will define $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ to be such a suitable poset (a space of anti-chains), and call it the 'arena of global positions' of $\mathfrak{D}$. Then, the concept of arena is formalized in the following way:

Definition 2 (Arena of Global Positions). *Given a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, the poset $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$ is its 'arena of global positions', where $\mathcal{A}$ is the set of anti-chains of $\mathfrak{D}$ and $E \leq_{\mathcal{A}} D$ iff $\downarrow E \subseteq \downarrow D$, for all anti-chains of $\mathfrak{D}$.*

The reader acquainted with partial order models of concurrency, in particular with event structures, may have noticed that the poset of anti-chains defined here is similar to the domain of representative elements of a *prime* event structure [15]—and therefore also to the set of states or markings in a *safe* Petri net.

Definition 3 (Global Strategies). *Let $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ be a board. Given two subsets of indices $\mathrm{K}_\forall$ of $\mathbb{K}_\forall$ and $\mathrm{K}_\exists$ of $\mathbb{K}_\exists$, and hence, two sets of local strategies $\Lambda^{\forall}_{\text{Ↄ}}$ and $\Lambda^{\exists}_{\text{Ↄ}}$ for Adam and Eve, let the closure operators $\partial_\forall : \mathcal{A} \rightarrow \mathcal{A}$ and $\partial_\exists : \mathcal{A} \rightarrow \mathcal{A}$ on the poset $\mathbb{A} = (\mathcal{A}, \leq_{\mathcal{A}})$, where $\mathbb{A}$ is the arena of global positions associated with $\mathfrak{D}$, be the 'global strategies' for Adam and Eve defined as follows:*

$$\partial_\forall(D) \stackrel{\text{def}}{=} \mathtt{max} \bigcup_{d \in D, j \in \mathrm{K}_\forall} \{\lambda^j_\forall(d) \mid \mathtt{dfn}(\lambda^j_\forall, d)\}$$
$$\partial_\exists(D) \stackrel{\text{def}}{=} \mathtt{max} \bigcup_{d \in D, i \in \mathrm{K}_\exists} \{\lambda^i_\exists(d) \mid \mathtt{dfn}(\lambda^i_\exists, d)\}$$

where $D \subseteq \mathcal{D}$ is a global position of $\mathfrak{D}$ (and therefore an element of $\mathcal{A}$) and `max` *is the 'maximal elements' set operation, which is defined as usual.*

The reasons why $\partial_\forall$ and $\partial_\exists$ are closure operators are as follows: extensiveness is given by the identity local strategies $\lambda^1_\forall$ and $\lambda^1_\exists$ and monotonicity and idempotency are inherited from that of local strategies and ensured by `max`. Note that `max` is needed for two reasons: because (1) a global position must be an anti-chain of local ones and (2) only representative elements of $\mathbb{A}$ should be considered.

Now, the dynamics of a game is given by the interaction between the players (together with an external environment $\amalg$ which is enforced to be deterministic by dsync—the property on boards and synchronization points given before).

Definition 4 (Rounds and Composition of Strategies). *Let a '$(\exists \circ \forall)$-round' be a global step of the game such that if $D \subseteq \mathcal{D}$ is the current global position of the game, $\partial_\exists$ is the strategy of Eve, and $\partial_\forall$ is the strategy of Adam, then the game proceeds first to an intermediate global position $D_{\exists\circ\forall}$ such that:*

$$\begin{aligned} D_{\exists\circ\forall} &= (\partial_\exists \circ \partial_\forall)(D) \\ &= \mathtt{max} \bigcup_{d \in D, i \in \mathrm{K}_\exists, j \in \mathrm{K}_\forall} \{(\lambda^i_\exists \circ \lambda^j_\forall)(d) \mid \mathtt{dfn}(\lambda^j_\forall, d) \wedge \mathtt{dfn}(\lambda^i_\exists, \lambda^j_\forall(d))\} \end{aligned}$$

and then to the next global position $D' = (D_{\exists\circ\forall} \setminus e^{\leftarrow}_{\mathtt{SP}}) \bigcup e^{\rightarrow}_{\mathtt{SP}}$, given by $\amalg$, where:

$$e^{\leftarrow}_{\mathtt{SP}} = \bigcup_{e \in D^{\rightarrow}_{\exists\circ\forall}} \{u \in e^{\leftarrow} \mid \mathtt{SP}(e) \wedge e^{\leftarrow} \subseteq D_{\exists\circ\forall}\}$$
$$e^{\rightarrow}_{\mathtt{SP}} = \bigcup_{e \in D^{\rightarrow}_{\exists\circ\forall}} \{v \in e^{\rightarrow} \mid \mathtt{SP}(e) \wedge e^{\leftarrow} \subseteq D_{\exists\circ\forall}\}$$

and call the transition from the global position D to D' a 'Ↄ-round' of the game.

This definition follows the intuition that in a logic game Eve must respond to any possible move of Adam; moreover, she has to do so in every local position.

Plays. The interaction between the strategies of Eve and Adam define a (possibly infinite) sequence of global positions $\{\bot\}, D_1, ..., D_k, ...$, and hence, a sequence of posets given by the union of the order ideals determined by each D_k. A play is any finite or infinite union of the elements of such posets. Formally, a play $\hbar = (\mathcal{H}, \leq_{\mathcal{D}})$ on a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ is a (possibly infinite) poset such that $\mathcal{H}$ is a downward-closed subset of $\mathcal{D}$. An example can be found in the appendix.

We say that a play can be finite or infinite, and closed or open; more precisely, a play is: *finite* iff all chains of $\hbar$ are finite; *infinite* iff $\hbar$ has at least one infinite chain; *closed* iff at least one of the terminal elements of $\mathfrak{D}$ is in $\mathcal{H}$; *open* iff none of the terminal elements of $\mathfrak{D}$ is in $\mathcal{H}$. This classification of plays is used in a further section to define in a concrete way what the winning sets of a game are. Since for any play $\{\bot\}, D_1, ..., D_k, ...$ the lower subset defined by a global position D_k always includes the lower subsets of all other global positions D_j such that $j < k$, then in a partial order setting any global position D determines a play $\hbar_D = (\mathcal{H}, \leq_{\mathcal{D}})$ on a board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ as follows (and let $\varGamma$ be the set of plays of a game): $\mathcal{H} = \bigcup\{e \in {\downarrow}d \mid d \in D\} = \bigcup_{d \in D}\{e \in \mathcal{D} \mid e \leq_{\mathcal{D}} d\}$.

Winning Sets and Strategies. The *winning* conditions are the rules that determine when a player has won a play and define the 'winning sets' for each player. Let $\mathcal{W} : \varGamma \rightarrow \varUpsilon$ be a *partial* function that assigns a winner $\exists, \forall \in \varUpsilon$ to a play $\hbar \in \varGamma$, and call it the winning conditions of a game. The winning sets are determined by those plays that contain a terminal element or represent infinite behaviour. On the other hand, *winning* strategies (which are global strategies) are defined as usual, i.e., as for games on graphs. Then, we have:

Definition 5. *$\Game = (\varUpsilon, \mathfrak{D}, \varLambda_{\mathfrak{D}}, \nabla, \mathcal{W}, \varGamma)$ is a 'Concurrent Logic Game' (CLG), where $\varUpsilon = \{\exists, \forall\}$ is the set of players, the $\bot$-bounded poset $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$ is a board, $\varLambda_{\mathfrak{D}} = \varLambda^{\exists}_{\mathfrak{D}} \uplus \varLambda^{\forall}_{\mathfrak{D}}$ are two disjoint sets of local strategies, $\nabla : \mathcal{D} \rightarrow \varUpsilon$ is a partial function that assigns players to local positions, and $\mathcal{W} : \varGamma \rightarrow \varUpsilon$ is a function defined by the winning conditions of $\Game$ over its set of plays $\varGamma$.*

A CLG is played as follows: Eve and Adam start by choosing, independently, a set of local strategies. The selection of local strategies is done indirectly by choosing the sets of indices $\mathrm{K}_{\forall} \subseteq \mathbb{K}_{\forall}$ and $\mathrm{K}_{\exists} \subseteq \mathbb{K}_{\exists}$. The only restriction (which we call '$\forall/\exists$-progress') when choosing the local strategies is that the resulting global strategy ∂, for either player, must preserve joins in the following way:

$$\begin{gathered}\forall d \in ({\uparrow}D \cap {\downarrow}\partial_{\forall}(D)), \text{ if } \mathtt{BP}(d) \wedge \nabla(d) = \forall \text{ then} \\ \forall a, b \in d^{\rightarrow}.\ \mathtt{sync}(a, b) \text{ implies } a, b \in {\downarrow}\partial_{\forall}(D).\end{gathered}$$

and likewise for Eve, changing $\partial_{\forall}$ for $\partial_{\exists}$ and the polarity given by ∇. The predicates $\mathtt{BP}$ and $\mathtt{sync}$ characterise, respectively, the 'branching points' of a poset and pairs of elements that belong to chains that synchronize; their definitions are: $\mathtt{BP}(d)$ iff $\mid d^{\rightarrow} \mid > 1$ and $\mathtt{sync}(a, b)$ iff ${\uparrow}a \cap {\uparrow}b \neq U$, for $U \in \{\emptyset, {\uparrow}a, {\uparrow}b\}$. This restriction—which avoids the undesired generation of trivial open plays where nobody wins—is necessary because a synchronization point can be played iff all elements of its preset have been played. Thus, this is a correctness condition.

3 Closure Properties

At least three closure properties are interesting: under dual games, under lower sub-boards, and its order dual, under upper sub-boards. But, before presenting the closures, let us give a simple, though rather useful, technical lemma, which helps ensure that in some sub-boards a number of functionals are preserved.

Lemma 1 (Unique Poset Prefixes). *Let D be a global position of a board $\mathfrak{D}$. There is a unique poset representing all plays containing D up to such position.*

Lemma 1 facilitates reasoning on CLG on posets as it implies that regardless of which strategies the players are using, if a global position D appears in different plays, then the 'poset prefixes' of all such plays, up to D, are isomorphic. Let us now study some of the closure properties the CLG model enjoys. Given a CLG $\eth$ played on a board $\mathfrak{D}$, let $\eth \Downarrow_{\mathfrak{B}}$ be the CLG defined from $\eth$ where $\mathfrak{B}$ is a sub-board of $\mathfrak{D}$ and the other components in $\eth$ are restricted to $\mathfrak{B}$.

Lemma 2 (Closure Under Filters). *Let $\eth$ be a CLG and D a global position of the board $\mathfrak{D}$ of $\eth$. The structure $\eth \Downarrow_{\mathfrak{B}} = (\Upsilon, \bot \oplus \mathfrak{B}, \Lambda_{\mathfrak{B}}, \nabla \Downarrow_{\mathfrak{B}}, \mathcal{W} \Downarrow_{\mathfrak{B}}, \Gamma \Downarrow_{\mathfrak{B}})$ is also a CLG where $\mathfrak{B}$ is the upper sub-board of $\mathfrak{D}$ defined by D.*

Where $\oplus$ is the 'linear sum' operator on posets. The order dual of this closure property is a closure under countable unions of (principal) ideals.

Lemma 3 (Closure Under Ideals). *Let $\eth$ be a CLG and D a global position of the board $\mathfrak{D}$ of $\eth$. The structure $\eth \Downarrow_{\mathfrak{B}} = (\Upsilon, \mathfrak{B}, \Lambda_{\mathfrak{B}}, \nabla \Downarrow_{\mathfrak{B}}, \mathcal{W} \Downarrow_{\mathfrak{B}}, \Gamma \Downarrow_{\mathfrak{B}})$ is also a CLG where $\mathfrak{B}$ is the lower sub-board of $\mathfrak{D}$ defined by D.*

Remark 2. Lemmas 2 and 3 show that the filters of $\mathfrak{D}$ define the 'subgames' of $\eth$; also, that the ideals in $\mathfrak{D}$ can define a subset of the set of plays of Γ. Moreover, notice that games on infinite trees can be reduced to the particular case when D is always a singleton set and where two chains in the board never synchronise.

Since CLG will be used for verification, another useful feature is that of having a game closed under *dual* games, this is, a game used to check the dual of a given property over the same board—i.e., for the same system(s). Formally:

Definition 6 (Dual Games). *Let $\eth = (\Upsilon, \mathfrak{D}, \Lambda_{\mathfrak{D}}, \nabla, \mathcal{W}, \Gamma)$ be a CLG. The dual game $\eth^{op}$ of $\eth$ is $(\Upsilon, \mathfrak{D}, \Lambda_{\mathfrak{D}}, \nabla^{op}, \mathcal{W}^{op}, \Gamma)$, such that for all $d \in \mathcal{D}$ and $\hbar \in \Gamma$:*

- *if $\nabla(d) = \exists$ (resp. $\forall$) then $\nabla^{op}(d) = \forall$ (resp. $\exists$), and*
- *if $\mathcal{W}(\hbar) = \exists$ (resp. $\forall$) then $\mathcal{W}^{op}(\hbar) = \forall$ (resp. $\exists$).*

Moreover, let $\mathfrak{I}$ be a class of CLG where for all $\eth \in \mathfrak{I}$ there is a dual game $\eth^{op} \in \mathfrak{I}$. Then, we say that $\mathfrak{I}$ is closed under dual games.

Lemma 4 (Closure Under Dual Games). *Let $\mathfrak{I}$ be a class of CLG closed under dual games. If Eve (resp. Adam) has a winning strategy in $\eth \in \mathfrak{I}$, then Adam (resp. Eve) has a winning strategy in the dual game $\eth^{op} \in \mathfrak{I}$.*

Lemma 4 does not imply that CLG are determined as the existence of winning strategies has not yet been ensured; let alone the guarantee that finite and open plays in which $D \rightarrow D'$ and $D = D'$ hold are not possible, as this implies that the game is undetermined. Call 'stable' a play where $D \rightarrow D'$ and $D = D'$ hold.

Another condition that is necessary, though not sufficient, for a game to be determined is that all plays that are not finite and open have a winner. This is ensured by requiring $\mathcal{W}$ to be *complete*. We say that $\mathcal{W}$ is complete iff it is a total function on the subset of plays in Γ that are not finite and open.

Lemma 5 (Unique Winner). *Let $\mathfrak{J}$ be a class of CLG closed under dual games for which plays that are stable, finite and open do not occur. If the $\mathcal{W}$ in $\Game \in \mathfrak{J}$ is complete, then every play in $\Game$ and $\Game^{op}$ has a unique winner.*

Remark 3. If a class of games is closed under dual games and has a complete set of winning rules, then a proof of determinacy, which does not rely on Martin's theorem [14], can be given if the game is *sound* (where Adam is a correct falsifier). More importantly, games with these features must also be *complete* (where Eve is a correct verifier), and therefore *determined*. In this way one can reduce reasoning on games by building completeness and determinacy proofs almost for *free*!

Let us finish with a counter-example (Figure 1 in the appendix) that shows that CLG are undetermined. This motivates the definition of a semantic condition that, when satisfied, allows for the construction of a determined CLG.

Proposition 1. *CLG are undetermined in the general case.*

4 Metatheorems for Systems and Property Verification

As a CLG model can be seen as a logic game representation of a verification problem (cf. [5]), then let $\Game_{\mathrm{P}}$ be the CLG associated with a decision problem $\mathrm{V(P)}$, for a given problem P, and $\mathfrak{J}$ the class of CLG representing such a decision problem. We say that $\mathrm{V(P)}$ holds iff such a decision problem has a positive answer, and fails to hold otherwise; then $\mathrm{V(P)}$ is used as a logical predicate.

As usual, $\Game_{\mathrm{P}}$ is correct iff Eve (resp. Adam) has a winning strategy in $\Game_{\mathrm{P}}$ whenever $\mathrm{V(P)}$ holds (resp. fails to hold). Let a 'local configuration' of $\Game_{\mathrm{P}} \in \mathfrak{J}$ be a local position and a 'global configuration' an anti-chain of local positions. Moreover, a 'true/false configuration' is a configuration from which Eve/Adam can win. A global configuration is logically interpreted in a conjunctive way; then, it is true iff it only has true local configurations, and false otherwise.

In order to show the correctness of the family of games $\mathfrak{J}$, in this abstract setting, we need to make sure that the CLG $\Game_{\mathrm{P}}$ associated with a particular verification problem $\mathrm{V(P)}$ has some semantic properties, which are given next.

Definition 7 (Parity/co-Parity Condition). *Let $(A, \leq_A)$ be a poset indexed by a finite subset of $\mathbb{N}$, $\overrightarrow{a}$ be a sequence of elements of A whose order respects $\leq_A$ and downward-closure, and $f^{\omega}_{\mathtt{min}} : A^{\omega} \rightarrow \mathbb{N}$ be a function that characterizes the minimum index that appears infinitely often in $\overrightarrow{a}$. Then, $(\{b \in A \mid b \in \overrightarrow{a}\}, \leq_A)$ is a poset definable by a 'Parity/co-Parity condition' iff $f^{\omega}_{\mathtt{min}}(\overrightarrow{a})$ is even/odd.*

Property 1 (ω-Symmetry: bi-complete ω-regularity). *A family $\mathfrak{J}$ of CLG has Property 1 and is said to be ω-symmetric (bi-complete and ω-regular), iff:*

1. *$\mathfrak{J}$ is closed under dual games;*
2. *for all $\Game_{\mathrm{P}} \in \mathfrak{J}$ we have that $\Game_{\mathrm{P}}$ has a complete set of winning conditions;*
3. *the winning set given by those plays such that $\mathcal{W}(\hbar) = \exists$, i.e., those where Eve wins, is definable by Büchi/Rabin/Parity conditions.*[1]

An immediate consequence of the previous property is the following:

Lemma 6. *If a CLG $\Game_{\mathrm{P}}$ is ω-symmetric, then it also satisfies that the winning set given by those plays such that $\mathcal{W}(\hbar) = \forall$, i.e., those where Adam wins, is definable by co-Büchi/Streett/co-Parity conditions.*

Note that parts 1 and 3 of Property 1 are given by the particular problem to be solved. It is well known that several game characterisations of many verification problems have these two properties. On the other hand, part 2 is a design issue. From a more algorithmic viewpoint, Property 1 and Lemma 6 imply that:

Lemma 7. *The winning sets of Adam are least fixpoint definable; and dually, the winning sets of Eve are greatest fixpoint definable.*

Recall that Büchi and Rabin conditions can be reduced to a Parity one. Moreover, a Parity condition characterises the winning sets (and plays) in the fixpoint modal logic $\mathcal{L}_\mu$ [6] as follows: infinite plays where the smallest index that appears infinitely often is even/odd satisfy greatest/least fixpoints and belong to the winning sets of Eve/Adam. As in our setting plays are posets, the order is the one given by the board. The following semantic condition must hold too:

Property 2 (Local Correctness). *Let $\mathfrak{D}$ be the board of a CLG $\Game_{\mathrm{P}}$. If $d \in \mathcal{D}$ is a false configuration, then either $\nabla(d) = \exists$ and all next configurations are false as well or $\nabla(d) = \forall$. Dually, if $d \in \mathcal{D}$ is a true configuration, then either $\nabla(d) = \forall$ and all next configurations are true as well or $\nabla(d) = \exists$.*

The game interpretation of Property 2 reveals the mathematical property that makes a CLG logically correct. Property 2 implies that not only the local positions that belong to a player must be either true or false local configurations, but also those that belong to II, i.e., the joins of $\mathfrak{D}$. Then, truth and falsity must be transferred to those local positions as well, so that the statements "Eve must preserve falsity" and "Adam must preserve truth" hold. Formally, one needs to ensure that the following restriction (which we call '$\Game$-progress') holds:

$$\bigsqcup D \neq \emptyset \Rightarrow \bigsqcup_{d \in D} \partial_{\forall}(\{d\}) \neq \emptyset \qquad \text{and} \qquad \bigsqcup D \neq \emptyset \Rightarrow \bigsqcup_{d \in D} \partial_{\exists}(\{d\}) \neq \emptyset$$

where $\bigsqcup$ is the 'join operator' on posets; call 'live' a play that is not stable, finite and open, as well as games whose strategies only generate live plays. $\Game$-progress guarantees that only live plays and games—where truth and falsity are preserved—are generated. Moreover, it allow us to show the soundness and completeness of this kind of CLG. A simple technical lemma is still needed: a direct application of Lemma 5 using $\Game$-progress and Property 1 gives Lemma 8:

[1] Büchi and Rabin conditions are defined as expected [12] as well as their duals.

Lemma 8. *Every play of a live, ω-symmetric $\Game_{\mathrm{P}}$ has a unique winner.*

Theorem 1 (Soundness). *If* V(P) *fails to hold, Adam can always win* $\Game_{\mathrm{P}}$*.*

And, due to the properties of the game, we get completeness almost for free. Moreover, determinacy with *pure* winning strategies—a property not obvious for concurrent games—follows from the soundness and completeness results.

Theorem 2 (Completeness). *If* V(P) *holds, Eve can always win* $\Game_{\mathrm{P}}$*.*

Corollary 1 (Determinacy). *Eve has a winning strategy in* $\Game_{\mathrm{P}}$ *iff Adam does not have it, and vice versa.*

5 Algorithmic Applications and Further Work

Solving a CLG $\Game_{\mathrm{P}}$ using the approach we presented here requires the construction of a winning game $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ (and with it a winning strategy) for either player, according to Theorems 1 and 2. This is in general an undecidable problem since the board $\mathfrak{D}$ can be infinitely large. However, in many practical cases $\mathfrak{D}$ can be given a finite representation where all information needed to solve the verification problem is contained. Let us finish this section with the following result:

Theorem 3 (Decidability). *The winner of any CLG* $\Game_{\mathrm{P}}$ *can be decided in finite time if the board* $\mathfrak{D}$ *in* $\Game_{\mathrm{P}}$ *has finite size.*

Although Theorem 3 is not a surprising result, what is interesting is that several partial order models can be given a finite poset representation which, in a number of cases, can be *smaller* than their interleaving counterparts. Therefore, the decidability result may have important practical applications whenever the posets can be kept small. This opens up the possibility of defining new *concurrent* decision procedures for different verification problems.

Remark 4. The size of a board $\mathfrak{D}$ in a CLG model can be smaller than the unfolding $\mathfrak{U}$ of the interleaving structure representing the same problem. The exact difference, which can be exponential, depends on the degree of independence or concurrency in $\mathfrak{D}$, i.e., on the number of elements of those chains that are independent and therefore must be interleaved in order to get $\mathfrak{U}$ from $\mathfrak{D}$. Only experiments can tell if the CLG model can have an important practical impact.

Applications to Bisimulation and Model-Checking. The game framework described in this manuscript can be applied to solve, in a uniform way, the bisimulation and modal μ-calculus [6] model-checking problems. In particular, the induced decision procedures generalize those defined by Stirling in [16]. A detailed description of the two reductions can be found in [12] (Chapter 5).

Future Work. The work presented here can be extended in various ways. In particular, related to the author's previous work, we intend to study the expressivity and applicability of the CLG model with respect to logics and equivalences for so-called 'true concurrency', where interleaving approaches, either concurrent or sequential, are not semantically powerful enough. Results on this direction would allow us to give a concurrent alternative to the higher-order logic games for bisimulation and (local) model-checking previously presented in [11,13].

References

1. Abramsky, S.: Event domains, stable functions and proof-nets. Electronic Notes in Theoretical Computer Science 172, 33–67 (2007)
2. Abramsky, S., Melliès, P.A.: Concurrent games and full completeness. In: LICS, pp. 431–442. IEEE Computer Society, Los Alamitos (1999)
3. Alfaro, L.D., Henzinger, T.A.: Concurrent omega-regular games. In: LICS, pp. 141–154. IEEE Computer Society, Los Alamitos (2000)
4. Alfaro, L.D., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science 386(3), 188–217 (2007)
5. Benthem, J.V.: Logic games, from tools to models of interaction. In: Logic at the Crossroads, pp. 283–317. Allied Publishers (2007)
6. Bradfield, J.C., Stirling, C.: Modal mu-calculi. In: Handbook of Modal Logic, pp. 721–756. Elsevier, Amsterdam (2007)
7. Esparza, J., Heljanko, K.: Unfoldings - A Partial-Order Approach to Model Checking. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (2008)
8. Ghica, D.R.: Applications of game semantics: From program analysis to hardware synthesis. In: LICS, pp. 17–26. IEEE Computer Society, Los Alamitos (2009)
9. Girard, J.Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
10. Godefroid, P.: Partial-Order Methods for the Verification of Concurrent Systems - An Approach to the State-Explosion Problem. LNCS, vol. 1032. Springer, Heidelberg (1996)
11. Gutierrez, J.: Logics and bisimulation games for concurrency, causality and conflict. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 48–62. Springer, Heidelberg (2009)
12. Gutierrez, J.: On Bisimulation and Model-Checking for Concurrent Systems with Partial Order Semantics. Ph.D. thesis, University of Edinburgh (2011)
13. Gutierrez, J., Bradfield, J.C.: Model-checking games for fixpoint logics with partial order models. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 354–368. Springer, Heidelberg (2009)
14. Martin, D.A.: Borel determinacy. Annals of Mathematics 102(2), 363–371 (1975)
15. Nielsen, M., Winskel, G.: Models for concurrency. In: Handbook of Logic in Computer Science, pp. 1–148. Oxford University Press, Oxford (1995)
16. Stirling, C.: Bisimulation, modal logic and model checking games. Logic Journal of the IGPL 7(1), 103–124 (1999)
17. Walukiewicz, I.: A landscape with games in the background. In: LICS, pp. 356–366. IEEE Computer Society, Los Alamitos (2004)

Appendix

All proofs and some examples can be found in Chapter 5 of the author's PhD thesis [12].[2] Here we present only some selected ones which closely relate with the main technical results in the paper. We also include the proof of the closure under dual games since this lemma is the key property that makes the proof of completeness of the game

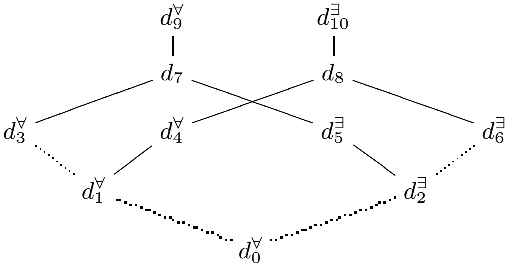
[2] This research was supported by an Overseas Research Studentship (ORS) award, a School of Informatics PhD scholarship of the University of Edinburgh, and the EPSRC Research Grant EP/G012962/1 'Solving Parity Games and Mu-Calculi'.

model rather short, even though we are working on a partial order setting. A simple example that illustrates how a CLG is played between Eve and Adam is also given in the proof of the following proposition.

Proposition 1. *CLG are undetermined in the general case.*

Proof. Neither player can have a winning strategy in the game presented in Figure 1 since Eve and Adam can enforce plays for which $\mathcal{W}$ is not defined (a stable, finite and open play). Notice that $\forall/\exists$-progress is not violated. □

Notice that the play presented in Figure 1 is the best that both players can do, since any other strategy they choose to play will lose against the strategy their opponent is currently playing in the example.



$\Game$-rounds: $\{d_0^{\forall}\} \rightarrow \{d_3^{\forall}, d_6^{\exists}\} \rightarrow \{d_3^{\forall}, d_6^{\exists}\} \rightarrow ...$

Fig. 1. Local positions are labelled with their polarities and the dotted lines are the player's moves. Here $\mathcal{W}(\downarrow D) = \exists$ iff $d_{10}^{\exists} \in D$ and $\mathcal{W}(\downarrow D) = \forall$ otherwise, for all global positions D containing a terminal element. The play is stable, finite and open.

Lemma 4 (Closure Under Dual Games). *Let $\mathfrak{J}$ be a class of CLG closed under dual games. If Eve (resp. Adam) has a winning strategy in $\Game \in \mathfrak{J}$, then Adam (resp. Eve) has a winning strategy in the dual game $\Game^{op} \in \mathfrak{J}$.*

Proof. Suppose that Eve has a winning strategy ∂_W in $\Game$. Since for all global positions in the game $\Game$ one has that the next global position is initially defined by $\partial_W \circ \partial_\forall$, then whenever Adam has to make a move in $\Game^{op}$ he can use the winning strategy ∂_W of Eve because for all $d \in D$, if $\nabla(d) = \exists$ then we have that $\nabla(d)^{op} = \forall$. However, notice that in each local position of the game board Adam must always "play first" both in $\Game$ and in $\Game^{op}$ because the global evolution of the game, which is determined by the rounds being played, is always defined by pairs of local strategies $\lambda_\exists^i$ and $\lambda_\forall^j$ such that $\lambda_\exists^i \circ \lambda_\forall^j(d)$, for any local position d, regardless of whether we are playing $\Game$ or $\Game^{op}$.

So, there are actually two cases: firstly, consider those $d \in D$, for any global position D, such that $\nabla(d) = \exists$. In this case $\nabla^{op}(d) = \forall$ and then in $\Game^{op}$ Adam can simply play Eve's strategy in $\Game$ at position d. The second case is that of those $d \in D$ such that $\nabla(d) = \forall$. In this case, $\nabla^{op}(d) = \exists$ and hence Adam can play d itself, and let Eve decide on the new local position d', for which, by hypothesis, Eve has a winning strategy in $\Game$ and the two previous cases apply again, though in a new round of the game; moreover, the behaviour at synchronization points, which are played deterministically by the environment II, remains as in $\Game$. In this way, Adam can enforce in $\Game^{op}$ all plays that Eve can enforce in $\Game$.

Finally, since for all such plays in $\mathit{\Gamma}$ it was, by hypothesis, Eve who was the winner, then Adam is the winner in all plays in $\Game^{op}$ as now for all $\hbar \in \mathit{\Gamma}$, one has that $\mathcal{W}^{op}(\hbar) = \forall$. The case when Adam has a winning strategy in $\Game$ is dual. □

Theorem 1 (Soundness). *If* $\mathrm{V(P)}$ *fails to hold, Adam can always win* $\Game_{\mathrm{P}}$.

Proof. We show that Adam can win all plays of $\Game_{\mathrm{P}}$ if $\mathrm{V(P)}$ fails to hold by providing a winning strategy for him. The proof has two parts: first, we provide a board where Adam can always win and show how to construct a game on that board, in particular, the local strategies in the game—and hence, a strategy for Adam; then, we show that in such a game Adam can always win by checking that his strategy is indeed a *pure* winning strategy.

Let $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ be a CLG on a poset $\mathfrak{B} = (\mathcal{B}, \leq_{\mathcal{D}})$, which is a subset of $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$, the initial board of the game. Let the set $\mathcal{B}$ be a downward-closed subset of $\mathcal{D}$ with respect to $\leq_{\mathcal{D}}$; the bottom element $\perp_{\mathcal{B}} = \perp_{\mathcal{D}}$ (where every play of the game starts) is, by hypothesis, a false configuration.

The construction of the board is as follows: $\mathfrak{B}$ contains only the winning choices for Adam (i.e., those that preserve falsity) as defined by the local correctness semantic property 2. After those elements of the poset have been selected, adjoin to them all possible responses or moves available to Eve that appear in $\mathfrak{D}$. Do this, starting from $\perp$, either infinitely often for infinite chains or until a terminal element is reached in finite chains. This construction clearly ensures that $\mathcal{B}$ is a downward-closed set with respect to $\leq_{\mathcal{D}}$. As in the proofs of Lemmas 2 and 3 (see [12] for further details), the polarity function ∇ for $\mathfrak{B}$ is as in $\mathfrak{D}$.

Using the constructions given in the proof of Lemma 2, one can define all other elements of $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$. In particular, the local strategies for Eve and Adam will be 'stable' closure operators;[3] based on Definition 1 such stable closure operators are completely defined once one has determined what the 'output' functions will be (the local positions d' in item 3) since the fixpoints are completely determined already in Definition 1. Then, each local strategy $\lambda^j_\forall$ for Adam and $\lambda^i_\exists$ for Eve—where $j \in \mathrm{K}_\forall \subseteq \mathbb{K}_\forall$ and $i \in \mathrm{K}_\exists \subseteq \mathbb{K}_\exists$, respectively—is defined as follows:[4]

$$\begin{aligned} \lambda^i_\exists(d) &= d \vee f^i_\exists(d) \\ \lambda^j_\forall(d) &= d \vee g^j_\forall(d) \end{aligned}$$

where:

$$\begin{aligned} \lambda^i_\exists(d) &= d && \text{, if } \mathtt{fix}_\exists(\lambda^i_\exists, d) \\ \lambda^i_\exists(d) &= f^i_\exists(d) && \text{, otherwise} \\ \lambda^j_\forall(d) &= d && \text{, if } \mathtt{fix}_\forall(\lambda^j_\forall, d) \\ \lambda^j_\forall(d) &= g^j_\forall(d) && \text{, otherwise} \end{aligned}$$

where each 'output' function $g^j_\forall$ necessarily preserves falsity and each output function $f^i_\exists$ must preserve truth (because $\mathfrak{B}$ was constructed taking into account Property 2). Moreover $\mathtt{fix}_\exists$ and $\mathtt{fix}_\forall$ are predicates that characterise the fixpoints of the local strategies for Eve and Adam in the following way:

$$\begin{aligned} \mathtt{fix}_\exists(\lambda_\exists, d) &\stackrel{\text{def}}{=} \mathtt{dfn}(\lambda_\exists, d) \text{ and } (\ \nabla(d) = \forall, \text{ or } \mathtt{SP}(d), \text{ or } d \not\rightarrow, \text{ or } d \rightarrow e \wedge \mathtt{SP}(e)\ ) \\ \mathtt{fix}_\forall(\lambda_\forall, d) &\stackrel{\text{def}}{=} \mathtt{dfn}(\lambda_\forall, d) \text{ and } (\ \nabla(d) = \exists, \text{ or } \mathtt{SP}(d), \text{ or } d \not\rightarrow, \text{ or } d \rightarrow e \wedge \mathtt{SP}(e)\ ) \end{aligned}$$

[3] Good references for stable maps on posets are [1,2] as well as some references therein.

[4] Recall that $i, j > 1$ as $\lambda^1_\forall$ and $\lambda^1_\exists$ are the identity local strategies of Adam and Eve.

where $d \in \mathcal{D}$, $\lambda_\forall \in \Lambda^{\forall}_{\mathfrak{D}}$, and $\lambda_\exists \in \Lambda^{\exists}_{\mathfrak{D}}$ in a game board $\mathfrak{D} = (\mathcal{D}, \leq_{\mathcal{D}})$.

However, since in $\mathfrak{B}$ all choices available to Eve were preserved, then the set of local strategies for Eve (i.e., $\Lambda^{\exists}_{\mathfrak{B}}$) can be safely chosen to be simply the same set of local strategies in $\mathfrak{D}$ (i.e., $\Lambda^{\exists}_{\mathfrak{D}}$); therefore, $\Lambda^{\exists}_{\mathfrak{B}} = \Lambda^{\exists}_{\mathfrak{D}}$ (because, formally, they play in $\mathfrak{D}$, even though Adam can prevent the positions in $\mathfrak{D} \setminus \mathfrak{B}$ to be reached) and $\Lambda^{\forall}_{\mathfrak{B}} \subseteq \Lambda^{\forall}_{\mathfrak{D}}$; moreover, the definition of global strategies immediately follows from this specification of local strategies as given by Definition 3 – of course, subject the restriction that any such global strategy must preserve the existence of joins in $\mathfrak{B}$ (progress restrictions). Finally, the sets of plays and winning conditions are defined from $\mathfrak{B}$ and the new sets of strategies as done in the proof of Lemma 2.

For the second part of this proof, let us show that the game $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ is winning for Adam, i.e., that he has a winning strategy. Then, let us analyse the outcome of plays to certify that he indeed wins all plays in such a game. First consider finite plays, which must be closed because all valid strategies must preserve the existence of joins. All such plays have a global position D_f which contains at least one local position that is a terminal element of $\mathfrak{B}$. Due to Property 1 (part 2), all those plays are effectively recognised as winning for one of the players, in this case for Adam: since $\bot$ is a false configuration, Eve must preserve falsity, and Adam is only playing strategies that also preserve falsity, then D_f contains at least one local position d_f which also is a false configuration, and therefore D_f is a false configuration as well since it is interpreted conjunctively. As a consequence all finite plays are winning for Adam. The same argument also applies for infinite, closed plays. The final case is that of open, infinite plays.

The correctness of this case is shown by a transfinite induction on a well-founded poset of sub-boards of $\mathfrak{B}$; this technique generalizes the analysis of approximants of fixpoints on interleaving structures (i.e., on total orders) to a partial ordered setting. So, let $(\mathcal{O}, \leq_{\mathcal{O}})$ be the following partial order on sub-boards (i.e., posets):

$$\begin{gathered}\mathcal{O} = \{\, \Game \Downarrow_{\uparrow D} \mid D \text{ is a global position of } \mathfrak{B}\} \\ \Game \Downarrow_{\uparrow D} \leq_{\mathcal{O}} \Game \Downarrow_{\uparrow D'} \text{ iff } {\uparrow} D' \subseteq {\uparrow} D\end{gathered}$$

The relation $\leq_{\mathcal{O}}$ is clearly well-founded because all finite and infinite chains in the poset $(\mathcal{O}, \leq_{\mathcal{O}})$ have $\bot_{\mathcal{O}} = \Game \Downarrow_{\uparrow \bot_{\mathcal{B}}} = \Game \Downarrow_{\mathfrak{B}}$ as their bottom element. Since any particular play in the game corresponds to a chain of $(\mathcal{O}, \leq_{\mathcal{O}})$, then let us also define a valuation $[\![\cdot]\!] : \mathcal{O} \rightarrow \{\textbf{true},\textbf{false}\}$ and a total order on the sub-boards (i.e., posets), and therefore subgames, associated with $\mathfrak{B}$. Let $\hbar$ be any open, infinite play (an infinite chain of $(\mathcal{O}, \leq_{\mathcal{O}})$) and let $\alpha, \varpi \in \mathbb{O}\mathrm{rd}$ be two ordinals, where ϖ is a limit ordinal. Then:

$$\begin{array}{lll} [\![\hbar^0]\!] & = [\![\bot_{\mathcal{O}}]\!] & \text{(the base case)} \\ [\![\hbar^{\alpha+1}]\!] & = [\![\rightarrow_{\mathcal{O}} (\hbar^\alpha)]\!] & \text{(the induction step)} \\ [\![\hbar^{\varpi}]\!] & = [\![\bigcup_{\alpha<\varpi} (\hbar^\alpha)]\!] & \text{(because } \varpi \text{ is a limit ordinal)} \end{array}$$

where $\rightarrow_{\mathcal{O}}$ is the accessibility relation of $\leq_{\mathcal{O}}$ restricted to the elements of the chain $\hbar$. Then, for Adam, we have the following:

$$\begin{array}{lll} [\![\bot_{\mathcal{O}}]\!] & = \textbf{false} & \text{(by hypothesis, } \bot_{\mathcal{O}} \text{ is a false configuration)} \\ [\![\rightarrow_{\mathcal{O}} (\hbar^\alpha)]\!] & = [\![\hbar^\alpha]\!] & \text{(due to Property 2, } \rightarrow_{\mathcal{O}} \text{ preserves falsity)} \\ [\![\bigcup_{\alpha<\varpi} (\hbar^\alpha)]\!] & = \bigvee_{\alpha<\varpi} \; [\![\hbar^\alpha]\!] & \text{(because due to Lemma 7,} \\ & & \text{Adam's winning sets are least fixpoint definable)} \end{array}$$

Due to the principle of (transfinite) fixpoint induction, the result holds for all ordinals, and therefore for all global positions of any open, infinite play. Note that we can actually repeat this analysis for all ordinals $\beta < \alpha$ (and thus for all global positions), due to Property 1 (part 3), since winning configurations, and hence winning sets, are fixpoint definable. But, since ordinals are well-founded such a process of checking subgames and open, infinite plays always terminates regardless of which α one chooses. Hence, there can be neither a D nor a game $\Game_{\mathrm{P}} \Downarrow_{\perp_{\mathcal{B}} \oplus \uparrow D}$ where Eve wins.

As she cannot win any finite or infinite play in $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$, and due to Lemma 8 all plays have a unique winner, Adam's strategy is indeed a winning strategy in $\Game_{\mathrm{P}}$; more precisely, it clearly is a *pure* winning strategy. Then, one can ensure that If $\mathrm{V}(\mathrm{P})$ fails to hold then Adam can win all plays of $\Game_{\mathrm{P}}$. □

A similar proof can be given to show the completeness of the game. Nevertheless, due to the properties of the game (notably, the closure under dual games), we can get the proof of completeness almost for free!

Theorem 2 (Completeness). *If* $\mathrm{V}(\mathrm{P})$ *holds, Eve can always win* $\Game_{\mathrm{P}}$*.*

Proof. Due to Property 1 (part 1) there exists a dual CLG $\Game_{\mathrm{P}}^{op}$ for the dual verification problem $\mathrm{V}(\mathrm{P}^{op})$ of $\mathrm{V}(\mathrm{P})$ such that $\mathrm{V}(\mathrm{P}^{op})$ does not hold. And, due to Theorem 1 Adam has a winning strategy in the game $\Game_{\mathrm{P}}^{op}$ for the dual problem P^{op}. Therefore, due to Lemma 4 and Lemma 8, Eve can use the local strategies of Adam in $\Game_{\mathrm{P}}^{op}$ to be the unique winner of all plays $\hbar \in \mathit{\Gamma}$ of $\Game_{\mathrm{P}}$, and hence the existence of a winning strategy for Eve in $\Game_{\mathrm{P}}$ follows. □

Theorem 3 (Decidability). *The winner of any CLG* $\Game_{\mathrm{P}}$ *can be decided in finite time if the board* $\mathfrak{D}$ *in* $\Game_{\mathrm{P}}$ *has finite size.*

Proof. Since $\mathfrak{D}$, by hypothesis, has finite size, then there are finitely many sub-boards $\mathfrak{B}$, and consequently, finitely many subgames $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ that must be checked before constructing a winning one for either player. Moreover, constructing a particular game $\Game_{\mathrm{P}} \Downarrow_{\mathfrak{B}}$ either for Eve or Adam as described in the proofs of Theorems 1 and 2 can be effectively done also because $\mathfrak{D}$ is finite, as follows.

Firstly, since $\mathfrak{B}$ is finite there are finitely many different strategies for Eve and Adam. Moreover, since those strategies are closure operators in a finite structure, then their sets of closed elements eventually stabilize. As a consequence, there are only finitely many possible different plays (and game configurations), whose winner can always be checked—because the game is determined and its set of winning conditions is complete. Therefore, a winning strategy can be chosen from the set of strategies of the game by exhaustively searching such a set, simply by comparing it against all possible strategies of the other player. As we assume that Properties 1 and 2 hold, they need not be verified. □

Dynamic Epistemic Algebra with Post-conditions to Reason about Robot Navigation

Alexander Horn

Oxford University Computing Laboratory, Oxford, UK
alex.horn@gmail.com

Abstract. Dynamic epistemic algebra establishes Galois connections and quantales as a basis for reasoning about knowledge in multi-agent systems. To date, these algebraic-axiomatic methods have been restricted to a positive fragment of dynamic epistemic logic with communication events only. This paper proposes Boolean algebraic extensions which overcome these limitations by generalizing dynamic epistemic algebra to scenarios where events can change facts in form of post-conditions. As an application of the new algebraic treatment of post-conditions, we devise and solve a topological map-based robot navigation example for which current axiomatics are insufficient.

1 Introduction

Networks such as the Internet enable autonomous agents to communicate with each other. Such group communication can be understood as information flow. The formal logical analysis of information flow in multi-agent systems is possible with dynamic epistemic logic. Dynamic epistemic logic is sound and complete with respect to relational semantics [4]. These relational semantics trace back to Kripke semantics [14] which shape modal logic (e.g. [5]). Similar to modal logic, traditional dynamic epistemic logic requires fixed valuations where facts never change [24]. To account for factual changes, recent research extended dynamic epistemic logic with post-conditions [25], [11], [21], [23]. The resulting logical expressiveness has found application in the study of market simulations, such as the trade on commodities [22], and the reasoning about information across covert channels [26]. However, relational semantical proofs can be tedious to formalize as agents interact frequently.

In contrast, algebraic semantics of dynamic epistemic logic abstract parts of the traditional relational structures by characterizing the dynamic nature of knowledge in terms of Galois connections and quantales [20], [3]. These mathematical structures tend to make proofs about information flow in multi-agent systems more perspicuous. However, only multi-agent systems in which facts never change are supported [18], [8]. In response, subsequent changes to the algebra aimed at increased flexibility with a converse dynamic modality [17]. However, this converse modality excludes event composition and it formalizes only learning of agents without initial knowledge [13].

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 161–175, 2011.

To overcome these limitations, this paper proposes a novel algebraic abstraction inspired by the relational semantical extensions of dynamic epistemic logic with post-conditions (e.g. [25], [21], [23]). The novelty is a dynamic epistemic algebra with Boolean algebraic structures and post-conditions for events in a quantale. The former applies to the negation of beliefs (i.e. disbeliefs). The latter enables reasoning about knowledge when events can change facts. As an application of factual changes, I encode the movements of a robot as events with post-conditions. This encoding aligns with topological maps studied in the robotics literature (e.g. [15]). Moreover, it broadens the application of dynamic epistemic logic by yielding a sensor-based robot navigation strategy which demonstrates the formalization and analysis of scenarios in which agents explore their surrounding including explorations from a known initial location.

2 Robot Navigation with Topological Maps

In order to explain the problem, we start with a directed graph $G = (V, E)$. Graph G is called a **topological map** if each vertex corresponds to a distinct **location** in an environment, such as an office, and edges denote **movements** between locations. Since topological maps are abstractions of the real world, locations could be understood as "distinctive places" which correspond to reference points in the environment [15].

Suppose there is an autonomous robot who knows the topological map of its surrounding. The objective of the robot is to determine its exact location. This objective could require the robot to move about in order to discover more information. During this exploration, the assumption is that the robot can detect all available movements at its current location. When the robot performs one of these movements, we assume that it always reaches the next location according to the topological map.

As an example of robot navigation, consider the topological map in Figure 1 devised by Phillips [18]. Let the robot's initial location be p_2. Since locations p_1 and p_2 are identical in terms of available movements (a and b), the robot considers the possibility of being in either location. Similarly, if it were to start at location p_3, it would not know if it would be in p_3 or p_4. However, as soon as it moves from its unknown initial location p_2 via movement a, it gains knowledge of its position! That is, the robot learns that it has arrived at location p_3. It rules out the possibility of having reached location p_4 because an a movement

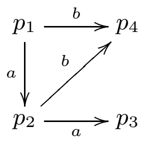


Fig. 1. Topological map in which the movement from location p_2 to p_3 reduces the uncertainty of the agent about its position [18], [8]

does not lead to it. Moreover, even though an a movement leads to location p_2, the robot eliminates this possibility as well because it can detect neither a nor b movements once it arrives at location p_3.

Noteworthy, these conclusions cannot be drawn with the original dynamic epistemic algebra [18]. Therefore, our goal is to algebraically formalize and precisely analyze how movements change the robot's uncertainty about locations.

Unlike previous algebraic approaches [18], [8], [7], [17], this paper aims at robot navigation with a more general solution where a robot has a sensor. This sensor-based solution can be described by the following algorithm [13]:

1. The sensor informs the robot of possible movement options (if any).
2. After choosing one of these movements, the robot relocates accordingly.
3. In the meantime, the robot uses its knowledge of the topological map to determine the set of possible destinations.
4. Once the robot arrives at one of these destinations, the sensor probes the environment and broadcasts the set of available movements.
5. By listening to this broadcast, the robot combines the information from the two previous steps to update its uncertainty about its current location.

In fact, this solution appeals to other problems in which agents need to observe changes in their surrounding to learn information. To emphasize this point, we simply represent the fourth step as an honest public announcement by the sensor [13]. In an epistemic setting, the challenge is the relocation of the robot because it causes factual changes.

Even though recent axiomatic extensions of dynamic epistemic logic [23] apply to robot navigation scenarios [13], such Hilbert-style deductions are not necessarily constructive in the sense that they explain the intuition behind machine learning ([13] gives a more thorough comparison between the logical and algebraic approaches). In Figure 1, for example, the proof that the robot learns its destination when moving from location p_2 to p_3 could consist of tautologies such as $\vdash [(p_3 \vee p_4)!]\Box_\alpha p_3 \leftrightarrow (p_3 \vee p_4) \to \Box_\alpha(p_4 \to p_3)$. Intuitively, after the honest public announcement by the sensor, the agent believes that it is in location p_3 if and only if it is neither in p_3 nor in p_4, or it believes that its presence at location p_4 implies that it is at location p_3. To avoid such indirection, the next section develops a dynamic epistemic algebra with post-conditions which conceptualizes learning as the change of an agent's uncertainty about locations.

3 Dynamic Epistemic Algebra with Post-conditions

The requirement is to analyze both information flow and factual changes in multi-agent systems. The former has been studied in terms of Galois connections and quantales [2], [20], [3]. The latter is the main contribution of this section. Both features require some level of familiarity with lattice theory (e.g. [9]).

To capture factual changes in an agent's surrounding, we start by defining a countable set of facts as a basis for real world information. Note that the separate treatment of logical propositions deviates from earlier intuitionistic algebraic approaches [20], [3].

Definition 1. *Define the tuple* $\mathcal{P} = \langle P; \vee, \wedge, \neg, \bot, \top \rangle$ *to be a Boolean algebra where* P *is the set of atoms which we call* ***facts****.*

Facts are atomic sentences which are either true or false. By definition ([1]), the Boolean algebra of facts satisfies classical propositional logic properties such as De Morgan's laws. For example, $\neg(p \wedge q) = \neg p \vee \neg q$ for all $p, q \in \mathcal{P}$.

Definition 2. *A* ***unital quantale*** $\langle E, \bigvee, \bullet, 1 \rangle$ *is a sup-lattice* E*, equipped with a monoid operation* $- \bullet -\colon E \to E$ *and the identity element* 1 *such that*

$$\left(\bigvee_j e_j\right) \bullet e = \bigvee_j (e_j \bullet e) \quad \textit{(jql)} \qquad e \bullet \left(\bigvee_i e_i\right) = \bigvee_i (e \bullet e_i) \quad \textit{(jqr)}$$

In other words, rule (jql) together with rule (jqr) define the monoid to preserve arbitrary joins in both arguments. Intuitively, elements in the quantale are interpreted as events ordered by determinism where $e \vee f$ corresponds to the non-deterministic choice of events [3]. Multiplication in the quantale can be interpreted as sequential event composition [3]. For example, $e \bullet f$ denotes the sequence of events where e happens before event f. Thus, multiplication in the quantale is defined to be noncommutative. Furthermore, quantales are resource-sensitive [20]. For example, $e \not\leq e \bullet e$ and $e \bullet e \not\leq e$ means that the repetition of the same event can be fundamentally different from its single occurrence as the repetitive group interrogation in the Muddy Children Puzzle illustrates (e.g. [10]). Since the quantale is unital, $e \bullet 1 = 1 \bullet e = e$ for all $e \in E$, where the identity element is the **unit event** which does nothing. Notice that the unit event is unrelated to the least or greatest element in the lattice. Apart from the unit event, all other events in the quantale have some sort of effect. This notion of 'change' leads to the definition of a module [20], [3]:

Definition 3. *Let* E *be a quantale. A* ***module*** *over* E *is a sup-lattice* M *with a function* $- \odot -\colon M \times E \to M$ *such that, for all* $m \in M$ *and* $e, f \in E$*,*

$$m \odot \left(\bigvee_i e_i\right) = \bigvee_i (m \odot e_i) \quad \textit{(je)} \qquad \left(\bigvee_j m_j\right) \odot e = \bigvee_j (m_j \odot e) \quad \textit{(jm)}$$

$$(m \odot e) \odot f = m \odot (e \bullet f) \quad \textit{(co)} \qquad m \odot 1 = m \quad \textit{(unit)}$$

Elements in the module embody 'possible worlds' [10], [2]. For this reason, these elements are called **states**. The dynamic nature of multi-agent systems is captured by the module operation $- \odot -$ which reflects changes of beliefs [3]. Since rule (je) and (jm) define this operation to be join-preserving in both arguments, it is monotonic (e.g. [9]) if either argument is fixed. The remaining equalities axiomatize event composition and the unit event [20], [3], [13].

Noteworthy, the pair of quantales and a module without modal operators has been previously studied with denotational models of concurrent processes [1]. Its connection with dynamic epistemic logic has been realized in [20] with a focus on intuitionistic logic where traditional Boolean laws cannot be generally applied. We start to address this limitation by refining the definition of a module to a complete Boolean lattice. Recall that a complete Boolean lattice is the same as a complemented distributive lattice where every subset has a supremum.

Definition 4. *A **Boolean module** is a module whose lattice is a complete Boolean lattice. In general, it is an infinite lattice [13, pp. 62f.].*

Similar to the relational semantics [25], [11], [21], [23], events in the quantale have pre-conditions and post-conditions [13, pp. 51–65]. The former is intrinsic to the update operation: if the pre-condition of an event e is unsatisfied in a state m, then $m \odot e = \bot$. The latter, however, requires the association of facts with elements in the Boolean module [13] because factual and epistemic changes evolve independently. This leads to the definition of the assignment operator:

Definition 5. *Let $\mathcal{P}$ be a Boolean algebra of facts. Let M be a Boolean module. Define the **assignment** to be a Boolean homomorphism $-^* : \mathcal{P} \to M$.*

Since the new assignment operator maps logical propositions to states, it shares much in common with valuations such as in modal logic, for example. The difference is that the algebraic association of states with facts is accomplished by the partial order relation on the complete Boolean lattice of the module. Similar to valuations, for more complex expressions such as $(p \wedge \neg q)^*$, we can conclude that it is equal to $p^* \wedge \neg q^*$ because the assignment operator is defined to be a Boolean homomorphism. This Boolean homomorphism builds the basis for the algebraic specification of post-conditions which we define next.

Definition 6. *Let $\mathcal{P}$ be a Boolean algebra of facts. Let E be a quantale and M be a Boolean module over E. Let $p \in \mathcal{P}$, $m \in M$ and $e \in E$. We say state m **satisfies** proposition p if $m \leq p^*$. We call proposition p the **post-condition** of event e in state m if $m \odot e$ satisfies p.*

Intuitively, $m \leq p^*$ means that in state m proposition p is true. We refer to such an inequality as **satisfaction relation**. By definition (5), satisfaction relations such as $m \leq (p \wedge \neg q)^*$ and $m \leq p^* \wedge \neg q^*$ are equivalent. As special cases of satisfaction relations, post-condition specifications are inequalities of the form $m \odot e \leq p^*$. Intuitively, such inequalities assert that proposition p is true after the update of a state m with an event e. Noteworthy, these updates can capture factual changes. Of course, it could also be the case that state m does not even satisfy the pre-condition of event e in which case the inequality $m \odot e = \bot \leq p^*$ is vacuously true. Such vacuous conditions are in accordance with the meta-logical treatment of pre-conditions in the relational approach where the outcome of an event is undefined if its pre-condition has not been satisfied [24]. Thus, post-condition specifications tend to be much more concise in both models without non-trivial cases. The introduction of post-conditions, however, warrants the formal differentiation between epistemic events and the new class of non-epistemic events.

Definition 7. *Let $\mathcal{P}$ be a Boolean algebra of facts. Let E be a quantale and M be a Boolean module over E. Let $e \in E$. Then, e is called an* ***epistemic event*** *if $p^* \odot e \leq p^*$ for all $p \in \mathcal{P}$. Otherwise, we call the event* ***non-epistemic****.*

In other words, if there exists $p \in \mathcal{P}$ such that $p^* \odot e \not\leq p^*$, then e is called a non-epistemic event. Informally, non-epistemic events can change the propositional assignment of the state being updated, whereas epistemic events preserve all propositional assignments. The latter corresponds to 'atomic permanence' in proof systems for dynamic epistemic logic without factual changes [4]. In fact, the definition of an epistemic event is also similar to the definition of **stable facts** in the intuitionistic approach [20], [3]. However, our approach achieves a clearer separation between stable facts and epistemic propositions by virtue of the additional assignment operator introduced as part of definition (5).

Definition 8. *A* ***Boolean system*** *is a quadruple consisting of a Boolean algebra with set of facts P together with a quantale E, a Boolean module M over E and an assignment. We write $\langle P, M, E, \odot, ^* \rangle$ for a Boolean system.*

Finally, we augment the Boolean system with the lax sup-endomorphisms defined for static intuitionistic approaches to dynamic epistemic logic [2], [20], [3].

Definition 9. *Let $\mathcal{A}$ be a finite set of agents and $\alpha \in \mathcal{A}$. Let $u^E_\alpha \colon E \to E$ and $u^M_\alpha : M \to M$ be sup-endomorphisms. Define the pair $\left(u^E_\alpha, u^M_\alpha\right)$ to be a* ***lax sup-endomorphism of a Boolean system*** *$\langle P, M, E, \odot, ^* \rangle$ such that, for all $m \in M$ and $e, f \in E$,*

$$u^M_\alpha(m \odot e) \leq u^M_\alpha(m) \odot u^E_\alpha(e) \tag{uui}$$

$$u^E_\alpha(e \bullet f) \leq u^E(e) \bullet u^E_\alpha(f) \tag{uci}$$

$$1 \leq u^E_\alpha(1) \tag{usi}$$

Semantically, the function u^M_α encodes the uncertainty of agent α about states [3]. Similarly, u^E_α is an agent's α uncertainty about events [3]. Examples for both sup-endomorphisms and their interpretations appear in [20], [3], [13].

Clearly, uncertainties could change as agents interact with their environment. The epistemic effects of such interactions are modeled by the uncertainty update inequality (uci) which captures the notion of learning [20], [3], [13]. Secondly, (uci) relates the uncertainty about an event composition to the individual sequential events [20], [3]. Lastly, (usi) asserts that when no event occurs an agent must consider the possibility that, in fact, nothing has happened [20].

Notice that the assignment operator integrates the Boolean algebra $\mathcal{P}$ with these sup-endomorphisms. However, we treat expressions such as $u^M_\alpha(p^*)$ as proof-theoretic constants which cannot be simplified for propositions $p \in \mathcal{P}$. More accurately, the general assumption is that the exact value of $u^M_\alpha(p^*)$ is unknown. The rationale is similar to the argument that we would not generally compute $u^M_\alpha(\top)$ because it requires an explicit construction of the entire Boolean module. However, it turns out that we can gain much ground without knowing the exact value of $u^M_\alpha(p^*)$. For simplicity, we *could* assume that agents have total uncertainty about states which satisfy the proposition p. Of course, the claim is *not* that $u_\alpha^M(p^*) = \top$. A simple counterexample is $u_\alpha^M(\bot^*) = u_\alpha^M(\bot) = \bot$ by definition (5) and remark (1).

Remark 1. Recall that $\bot = \bigvee \emptyset$. By join-preservation of $- \odot -$, we conclude $m \odot \bot = \bot \odot e = \bot$ for all $m \in M$ and $e \in E$. Similarly, $u_\alpha^M(\bot) = \bot$.

Most notably, since u_α^M preserves arbitrary joins, we conclude that it has a Galois connection $u_\alpha^M(-) \dashv \Box_\alpha -$ (e.g. [9]). In fact, the right adjoint $\Box_\alpha -$ encodes the belief modality for agent α [20], [3]. Let $m \in M$ be a state, $e \in E$ be an event and $p \in \mathcal{P}$ be a proposition. By definition of Galois connection, $u_\alpha^M(m) \leq p^*$ if and only if $m \leq \Box_\alpha p^*$. In other words, if the uncertainty of the agent about state m includes only states which satisfy proposition p, then $\Box_\alpha p^*$ can be read as "agent α believes proposition p is true". Likewise, since $- \odot e$ preserves arbitrary joins, it has an adjoint denoted by $[e]-$. Recall that if the inequality $m \odot e \leq p^*$ is satisfied, then the post-condition of event e in state m is proposition p by definition (6). By adjunction $- \odot e \dashv [e]-$, this inequality holds if and only if $m \leq [e]p^*$. Intuitively, the expression $[e]p^*$ means that "after event e, proposition p holds". Of course, unless the pre-condition is actually satisfied, the inequality $m \leq [e]p^*$ is vacuously true because $m \odot e = \bot \leq p^*$.

4 Robot Movements as Post-conditions

This section exemplifies the algebraic treatment of post-conditions. For this purpose, we devise a robot navigation example based on the topological map shown in Figure 2. Notice that the robot can uniquely identify location p_1 because no other location has c movements. Similarly, location p_3 is unique. Therefore, the robot knows its destination when it moves from either of these known initial locations to any other. Recall that such valid conclusions cannot be generally drawn from algebraic semantics which require the converse dynamic modality [13]. In contrast, the extended dynamic epistemic algebra can be used to prove these and other scenarios [13].

Before we can illustrate this point, we must agree on a precise problem specification. For this purpose, we use the extended dynamic epistemic algebra defined in the previous section. Let $\mathcal{P}$ be a Boolean algebra where each fact in the set P represents a location of the robot according to the topological map. For example,

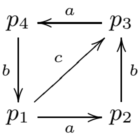


Fig. 2. Robot navigation example which demonstrates the algebraic treatment of post-conditions by proving that the robot knows its destination when it moves from a known initial location such as p_1 or p_3

the fact $p_1 \in P$ means that the robot is in location p_1. Since the robot can never be at two locations at the same time, we define $p_i \wedge p_j = \bot$ for all $p_i, p_j \in P$ such that $p_i \neq p_j$. Let M be a Boolean module such that for all facts $p_i \in P$ there exists an atom $m_i \in M$ where $m_i \leq p_i^*$. By definition (6), $m_i \leq p_i^*$ means that state m_i satisfies the fact that the robot is at location p_i. Since $m_i \neq \bot$, $p_i^* \neq \bot$ for all facts $p_i \in P$. Let movements between locations be events in the quantale E. Since one of the assumptions is that the robot has absolute certainty about its movements, $u_\alpha^E(e) := e$ for all $e \in E$. Define post-conditions by $p_i^* \odot e \leq p_j^*$, for all $p_i, p_j \in P$ and $e \in E$, such that p_j is the fact for the location which can be reached from location p_i with movement e. If such a movement is impossible, then $p_i^* \odot e = \bot$. Prior to any such movement, the uncertainty about initial locations is solely determined by available movements. Formally, we define $m_i \in M$ to be an **initial state** if m_i is an atom and $m_i \neq m_j \odot e$ for all $m_j \in M$ and $e \in E$ such that $e \neq 1$. The set of all such initial states is denoted by $I(M)$. Finally, define the uncertainty about an initial state m_i by $u_\alpha^M(m_i) := \bigvee\{m_j \in I(M) \mid \mathit{events}(m_i) = \mathit{events}(m_j)\}$ where $\mathit{events}(m_i)$ is the set of available movements at location p_i.

To develop an intuition for these definitions, reconsider the topological map in Figure 2. Since the robot can move from location p_1 to p_2 via movement a, the inequality $p_1^* \odot a \leq p_2^*$ is true. Furthermore, $u_\alpha^E(a) = a$ because the assumption is that the robot has absolute certainty about its movements. Finally, note that the robot has no uncertainty about the initial location p_1 because movement c is unique to it. Therefore, $u_\alpha^M(m_1) = m_1$ where $m_1 \in I(M)$ and $m_1 \leq p_1^*$.

Next, we use these definitions to prove that the robot has no uncertainty about its destination when moving from a known initial location such as p_1. Formally, $m_1 \leq [a]\Box_\alpha p_2^*$ where $m_1 \in I(M)$ and $m_1 \leq p_1^*$. The left side of the inequality corresponds to the robot's known initial location p_1. On the right side, the dynamic modality describes the robot's movement from location p_1 to p_2. After this movement, $\Box_\alpha p_2^*$ means that the agent believes that it is in location p_2. Before establishing this dynamic knowledge property, we prove that the robot, in fact, reaches location p_2 after starting from p_1.

Property 1. Let $m_1 \in M$ and $p_1, p_2 \in P$. Assume $m_1 \leq p_1^*$ and $p_1^* \odot a \leq p_2^*$. Then, $m_1 \odot a \leq p_2^*$.

Proof. By assumption, $m_1 \leq p_1^*$ and $p_1^* \odot a \leq p_2^*$. Since $- \odot a$ is monotonic, $m_1 \odot a \leq p_1^* \odot a$. By transitivity, $m_1 \odot a \leq p_2^*$.

Property (1) is a post-condition specification according to definition (6). It is independent from the robot's knowledge of the topological map. Since the next proposition, however, proves that the robot knows its destination, it also requires the additional assumption about the agent's uncertainty about its initial location p_1, i.e. $u_\alpha^M(m_1) = m_1$ where $m_1 \in I(M)$ and $m_1 \leq p_1^*$.

Proposition 1. *Assume that agent α knows the topological map in Figure 2. Let $m_1 \in I(M)$ where $m_1 \leq p_1^*$ and $p_1, p_2 \in P$. Then, $m_1 \leq [a]\Box_\alpha p_2^*$.*

Proof. *By the adjunction* $- \odot a \dashv [a]-$, *the claim is equivalent to* $m_1 \odot a \leq \Box_\alpha p_2^*$. *By the adjunction* $u_\alpha^M \dashv \Box_\alpha$, *this inequality is equivalent to* $u_\alpha^M(m_1 \odot a) \leq p_2^*$. *By (uui), it suffices to show* $u_\alpha^M(m_1) \odot u_\alpha^E(a) \leq p_2^*$. *By the topological map definition,* $m_1 \odot a \leq p_2^*$. *By property* (1), *the conclusion follows.*

The dynamic epistemic algebra with post-conditions also supports more sophisticated situations when the agent does not know its initial location [13]. Furthermore, the algebra integrates with communication events such as honest public announcements [13]. The combination of these features are sufficiently expressive, for example, for the sensor-based robot navigation strategy (p. 163) and the topological map in Figure 1 [13]:

Proposition 2. *Assume agent* α *knows the topological map in Figure 1 (p. 162). Let* $m_2 \in I(M)$ *such that* $m_2 \leq p_2^*$. *Then,* $m_2 \leq [a][(p_3^* \vee p_4^*)!]\Box_\alpha p_3^*$ *where* $(p_3^* \vee p_4^*)!$ *denotes the honest public announcement by the sensor.*

Proof. The proof appears in Appendix A.

5 Disbelief

Since previous algebraic semantics were built for intuitionistic logic [20], it has been difficult to express the negation of beliefs such as $\neg\Box_\alpha\phi$ because it required another Galois connection [20] without direct proof-theoretic applications [13]. In this section, we identify another Boolean algebraic characterization of the negation of agent's beliefs. Before we do this, the next definition suggests a more natural reading of an agent's negated beliefs.

Definition 10. *Let* M *be a Boolean module. Let* $\phi \in M$. *Define* ***disbelief*** *of an agent* α *about* ϕ *by* $\neg\Box_\alpha\phi$.

The next proposition contributes to the algebraic reasoning about disbeliefs.

Proposition 3. *Let* M *be a Boolean module. Let* $m, \phi \in M$ *be states. Then,* $m \leq \neg\Box_\alpha\phi$ *if and only if, for all* $x \in M$, $u_\alpha^M(x) \leq \phi$ *implies* $m \wedge x = \bot$.

Proof. The proof appears in Appendix B.

Proposition (3) relates disbeliefs to the greatest lower bound (infimum) of elements in the Boolean module. The next theorem proves a satisfying condition of disbelief for the special case in which one of these elements is an atom.

Theorem 1. *Let* M *be a Boolean module. Let* $m, \phi \in M$. *If* m *is an atom and* $u_\alpha^M(m) \not\leq \phi$, *then* $m \leq \neg\Box_\alpha\phi$.

Proof. The proof appears in Appendix B.

The next example demonstrates a proof about disbelief as a consequence of theorem (1).

Example 1. Consider a robot who knows the topological map in Figure 2 (p. 167). Let $m_2, m_4 \in I(M)$ be initial states and $p_2, p_4 \in P$ be facts. Assume $m_2 \leq p_2^*$ and $m_4 \leq p_4^*$. To show that the robot cannot clearly identify its initial location p_2, we must prove $m_2 \leq \neg\Box_\alpha p_2^*$. By definition of $I(M)$, both m_2 and m_4 are atoms. By theorem (1), it suffices to show that $u_\alpha^M(m_2) \not\leq p_2^*$. Since locations p_2 and p_4 have both only a movements, the agent considers the possibility of being in either initial location. Formally, $u_\alpha^M(m_2) = m_2 \vee m_4$. Since $m_4 \not\leq p_2^*$, we conclude that $u_\alpha^M(m_2) \not\leq p_2^*$. Therefore, $m_2 \leq \neg\Box_\alpha p_2^*$.

6 Conclusions

The main contribution of this paper is an algebraic framework for the formalization and analysis of agents which can observe and reason about changes in their environment. This achievement has been possible with the extension of dynamic epistemic algebra with post-conditions. For this purpose, the extended dynamic epistemic algebra integrates Galois connections, quantales and modules [20], [3] with an additional Boolean algebraic assignment operator. The significance of the assignment operator is threefold. Firstly, it achieves a clearer separation between stable facts and epistemic propositions. Secondly, it enables the algebraic abstraction of factual changes in form of events with post-conditions. Lastly, it unifies the algebraic treatment of both factual and epistemic changes under previously developed lax sup-endomorphisms. This unification eliminates the need for converse dynamic modalities (e.g. [18], [8], [7], [17]). Unfortunately, this elimination also reduces the expressiveness about temporal properties. In turn, however, the simplicity of dynamic epistemic algebra has been restored even for events which change facts. The algebraic specification of these events was shown to serve as a proof-theoretic basis for the encoding of robot movements.

7 Further Research

Future research could aim at the extension with fuzzy Galois connections [6]. Another research direction is the integration with Kleene algebra to simplify the algebraic characterization of common knowledge in resemblance to [16]. The resulting algebraic semantics could be a candidate for automated theorem provers (e.g. [12]). Such an implementation strategy could be compared and contrasted to epistemic model checking.

Acknowledgements. I am grateful to M. Sadrzadeh whose research questions sparked this paper. Her generous feedback improved the paper's presentation.

References

1. Abramsky, S., Vickers, S.: Quantales, observational logic and process semantics. Mathematical Structures in Computer Science 3(02), 161–227 (1993)
2. Baltag, A., Coecke, B., Sadrzadeh, M.: Algebra and sequent calculus for epistemic actions. In: Proceedings of the 2nd International Workshop on Logic and Communication in Multi-Agent Systems. Electronic Notes in Theoretical Computer Science, vol. 126, pp. 27–52 (2005)

3. Baltag, A., Coecke, B., Sadrzadeh, M.: Epistemic actions as resources. Journal of Logic and Computation 17, 555–585 (2007)
4. Baltag, A., Moss, L.S., Solecki, S.: The logic of public announcements, common knowledge, and private suspicions. In: Proceedings of TARK 1998, pp. 43–56. Morgan Kaufmann, San Francisco (1998)
5. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2001)
6. Bělohlávek, R.: Fuzzy galois connections. Math. Log. Q. 45, 497–504 (1999)
7. Philips, C., Precup, D., Panangaden, P., Sadrzadeh, M.: Reasoning about factual games using information updates. In: Proc. of 9th Conference on Logic and the Foundations of Game and Decision Theory (to appear)
8. Philips, C., Precup, D., Panangaden, P., Sadrzadeh, M.: An algebraic approach to dynamic epistemic logic. In: Proc. of 23rd International Workshop on Description Logics (2010)
9. Davey, B.A., Priestley, H.A.: Introduction to lattices and order, 2nd edn. Cambridge University Press, Cambridge (2002)
10. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
11. Herzig, A., De Lima, T.: Epistemic actions and ontic actions: A unified logical framework. In: Sichman, J.S., Coelho, H., Rezende, S.O. (eds.) IBERAMIA 2006 and SBIA 2006. LNCS (LNAI), vol. 4140, pp. 409–418. Springer, Heidelberg (2006)
12. Höfner, P., Struth, G.: Automated reasoning in kleene algebra. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 279–294. Springer, Heidelberg (2007)
13. Horn, A.: Reasoning about learning in robot navigation via algebraic dynamic epistemic logic. Master's thesis. University of Oxford (2010)
14. Kripke, S.A.: A semantical analysis of modal logic I: Normal modal propositional calculi. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9, 67–96 (1963)
15. Kuipers, B., Byun, Y.-T.: A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Journal of Robotics and Autonomous Systems 8, 47–63 (1991)
16. Möller, B.: Knowledge and games in modal semirings. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS/AKA 2008. LNCS, vol. 4988, pp. 320–336. Springer, Heidelberg (2008)
17. Panangaden, P., Sadrzadeh, M.: Learning in a changing world: an algebraic modal logical approach. In: Johnson, M., Pavlovic, D. (eds.) AMAST 2010. LNCS, vol. 6486, pp. 128–141. Springer, Heidelberg (2011)
18. Phillips, C.: An algebraic approach to dynamic epistemic logic. Master's thesis. McGill University (2009)
19. Rasiowa, H., Sikorski, R.: The mathematics of metamathematics. Państwowe Wydawn. Naukowe (1968)
20. Sadrzadeh, M.: Actions and Resources in Epistemic Logic. PhD thesis. Université du Québec à Montréal (2006)
21. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Inf. Comput. 204(11), 1620–1662 (2006)
22. van Ditmarsch, H.: The logic of pit. Synthese 149(2) (2006)
23. van Ditmarsch, H., Kooi, B.: Semantic results for ontic and epistemic change. In: Bonanno, G., van der Hoek, W., Wooldridge, M. (eds.) Logic and the Foundations of Game and Decision Theory, pp. 87–117. Amsterdam University Press, Amsterdam (2008)

24. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Springer, Heidelberg (2007)
25. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic epistemic logic with assignment. In: Proc. of AAMAS 2005, pp. 141–148. ACM, New York (2005)
26. van Ditmarsch, H., van Eijck, J., Wu, W.: One hundred prisoners and a lightbulb — logic and computation. In: Lin, F., Sattler, U. (eds.). AAAI Press, Menlo Park (2010)

Appendix A: Robot Navigation

This appendix proves proposition (2) (p. 169) which exemplifies how a robot learns its position as it moves from an *unknown* initial location. To help illustrate the relevance of the definitions in the precise analysis of epistemic and factual changes, the solution to the example is structured around property assertions.

Property 2. Let $m_1, m_2 \in I(M)$ such that $m_1 \leq p_1^*$ and $m_2 \leq p_2^*$. Assume $p_1^* \odot a \leq p_2^*$ and $p_2^* \odot a \leq p_3^*$. Then, $m_1 \odot a \leq p_2^*$ and $m_2 \odot a \leq p_3^*$

Proof. By assumption, monotonicity of $- \odot a$ and transitivity.

Property (2) is a post-condition specification according to definition (6). The first inequality specifies the post-condition of event a in state m_1 to be the fact that the robot reaches location p_2. Similarly, the post-condition of $m_2 \odot a$ is the fact for location p_3.

The next two properties aim at the formalization of the sensor's honest public announcement which is essential for the robot to learn its position.

Property 3. Let $\mathcal{P}$ be a Boolean algebra of facts P. Let $p_i, p_j \in P$ be distinct facts (i.e. $i \neq j$) such that $p_i \wedge p_j = \bot$. If $p_i^* \neq \bot$, then $p_i^* \not\leq p_j^*$.

Proof. Assume $p_i \wedge p_j = \bot$ and $p_i^* \neq \bot$. Assume $p_i^* \leq p_j^*$. Then, $p_i^* \wedge p_j^* = p_i^*$. Since $-^*$ is a Boolean homomorphism, $p_i^* \wedge p_j^* = (p_i \wedge p_j)^* = \bot^* = \bot = p_i^*$ contradicting the assumption. Hence, $p_i^* \not\leq p_j^*$.

Property 4. Let $\mathcal{P}$ be a Boolean algebra of facts P. Let $p_2, p_3, p_4 \in P$ be distinct facts. Then, $p_2^* \odot (p_3^* \vee p_4^*)! \leq \bot$ where $(p_3^* \vee p_4^*)!$ denotes the honest public announcement of the proposition $p_3 \vee p_4$.

Proof. By property (3), $p_2^* \not\leq p_3^*$ and $p_2^* \not\leq p_4^*$. Therefore, $p_2^* \not\leq p_3^* \vee p_4^*$. Hence, p_2^* does not satisfy the pre-condition of the honest public announcement.

The next property eliminates the possibility of the robot reaching location p_3 or p_4 from the initial location p_1 via movement a due to the truthful information sharing by the sensor.

Property 5. Let $m_1 \in I(M)$ with $m_1 \leq p_1^*$. Then, $m_1 \odot a \odot (p_3^* \vee p_4^*)! = \bot$.

Proof.

$$\begin{aligned}
m_1 \odot a &\leq p_2^* && \{\text{property } (2)\}\\
m_1 \odot a \odot (p_3^* \vee p_4^*)! &\leq p_2^* \odot (p_3^* \vee p_4^*)! && \{\text{monotonicity of } - \odot -\}\\
p_2^* \odot (p_3^* \vee p_4^*)! &\leq \bot && \{\text{property } (4)\}\\
m_1 \odot a \odot (p_3^* \vee p_4^*)! &\leq \bot && \{\text{transitivity}\}
\end{aligned}$$

Property 6. Let $m_2 \in I(M)$ with $m_2 \leq p_2^*$. Then, $m_2 \odot a \odot (p_3^* \vee p_4^*)! \leq p_3^*$.

Proof. Similar to property (5) except that $m_2 \odot a$ satisfies the pre-condition of the honest public announcement $(p_3^* \vee p_4^*)!$ because $m_2 \odot a \leq p_3^*$.

Finally, we prove the claim that the robot learns its destination after moving from location p_2 to p_3 via movement a and then listening to the honest public announcement by the sensor. Since this proof requires the assumption that the robot knows the topological map in Figure 1 (p. 162), we can gain additional clarity by stating the robot's uncertainty about its initial location p_2.

Property 7. Assume that agent α knows the topological map in Figure 1 (p. 162). Let $m_1, m_2 \in I(M)$ such that $m_1 \leq p_1^*$ and $m_2 \leq p_2^*$. Then, $u_\alpha^M(m_2) = m_1 \vee m_2$.

Proof. Since there are the same available movements at location p_1 and p_2, $\mathit{events}(m_1) = \mathit{events}(m_2)$. By definition of the uncertainty about initial states, we conclude that $u_\alpha^M(m_2) = m_1 \vee m_2$.

Proposition 4. *Assume agent α knows the topological map in Figure 1 (p. 162). Let $m_2 \in I(M)$ such that $m_2 \leq p_2^*$. Then, $m_2 \leq [a][(p_3^* \vee p_4^*)!]\Box_\alpha p_3^*$.*

Proof.

$$\begin{aligned}
m_2 &\leq [a][(p_3^* \vee p_4^*)!]\Box_\alpha p_3^* && \{\text{claim}\}\\
m_2 \odot a &\leq [(p_3^* \vee p_4^*)!]\Box_\alpha p_3^* && \{\text{gal}\}\\
(m_2 \odot a) \odot (p_3^* \vee p_4^*)! &\leq \Box_\alpha p_3^* && \{\text{gal}\}\\
u_\alpha^M((m_2 \odot a) \odot (p_3^* \vee p_4^*)!) &\leq p_3^* && \{\text{gal}\}\\
u_\alpha^M(m_2 \odot a) \odot u_\alpha^E((p_3^* \vee p_4^*)!) &\leq p_3^* && \{\text{uui}\}\\
\big(u_\alpha^M(m_2) \odot u_\alpha^E(a)\big) \odot u_\alpha^E((p_3^* \vee p_4^*)!) &\leq p_3^* && \{\text{uui}\}\\
((m_1 \vee m_2) \odot a) \odot (p_3^* \vee p_4^*)! &\leq p_3^* && \{\text{property } (7)\}\\
(m_1 \odot a \vee m_2 \odot a) \odot (p_3^* \vee p_4^*)! &\leq p_3^* && \{\text{jm}\}\\
m_1 \odot a \odot (p_3^* \vee p_4^*)! \vee m_2 \odot a \odot (p_3^* \vee p_4^*)! &\leq p_3^* && \{\text{jm}\}\\
m_2 \odot a \odot (p_3^* \vee p_4^*)! &\leq p_3^* && \{\text{property } (5)\}\\
p_3^* &\leq p_3^* && \{\text{property } (6)\}
\end{aligned}$$

The second, third and fourth proof lines appeal to the Galois connections for the a movement, the honest public announcement by the sensor and the belief modality respectively. The uncertainty update inequality is applied twice to reduce a more complex expression to a simpler one by separating the operands of the module operation. The resulting expression is in terms of the uncertainty about events, $u^E_\alpha(a)$ and $u^E_\alpha((p^*_3 \vee p^*_4)!)$, in addition to the uncertainty about the initial state. By definition of movements and honest public announcements, the uncertainty of both events is the identity function. By property (7) and property (2), the robot's uncertainty $u^M_\alpha(m_2)$ about its initial location p_2 lets it conclude that the subsequent a movement must lead either to location p_2 or p_3. After the honest public announcement by the sensor, the robot can eliminate the possibility of being in location p_2 by property (5). By property (6), it remains only the possibility of being in location p_3 proving the claim.

Appendix B: Disbelief

For the proof of proposition (3) (p. 169), we use the next lemma which appears in meta-mathematical discussions about Boolean entailment[1] [19]:

Lemma 1. *Let M be a Boolean module. Let $x, y \in M$. Then,*

$$x \wedge y = \bot \quad \textit{iff} \quad x \leq \neg y$$

The next lemma states the well-known result that every sup-homomorphism, which preserves arbitrary joins, has a right adjoint (e.g. [9]):

Lemma 2. *Let P and Q be ordered sets and $f\colon P \to Q$ be a function that preserves arbitrary joins. Then, there exists a function $g\colon Q \to P$ such that $f \dashv g$. In fact, $g(y) = \bigvee \{x \in P \mid f(x) \leq y\}$.*

The last lemma concerns the join of a set of elements in a complete lattice.

Lemma 3. *Let P be a complete lattice. Let $p \in P$ and $Q \subseteq P$. Then,*

$$\bigvee Q \leq p \quad \textit{iff} \quad x \leq p \textit{ for all } x \in Q$$

Proof. The proof is trivial by definition of least upper bound of the set Q.

The combination of these lemmas together with full distributivity of the Boolean module proves the algebraic characterization of an agent's disbeliefs.

Proposition 5. *Let M be a Boolean module. Let $m, \phi \in M$ be states. Then, $m \leq \neg\Box_\alpha\phi$ if and only if, for all $x \in M$, $u^M_\alpha(x) \leq \phi$ implies $m \wedge x = \bot$.*

Proof.

$$\begin{array}{llll} m \leq \neg\Box_\alpha\phi & \textit{iff} & m \wedge \Box_\alpha\phi = \bot & \{\textit{lemma } (1)\} \\ & \textit{iff} & m \wedge \bigvee\left\{x \in M \mid u^M_\alpha(x) \leq \phi\right\} = \bot & \{\textit{lemma } (2)\} \\ & \textit{iff} & \bigvee\left\{m \wedge x \in M \mid u^M_\alpha(x) \leq \phi\right\} = \bot & \{\textit{distributivity}\} \\ & \textit{iff} & (\forall x \in M)\, u^M_\alpha(x) \leq \phi \textit{ implies } m \wedge x = \bot & \{\textit{lemma } (3)\} \end{array}$$

What remains to show is theorem (1) as a special case of proposition (3).

[1] $\neg x \vee y = \top$ if and only if $x \leq y$ for all elements x, y in a Boolean algebra.

Theorem 2. *Let M be a Boolean module. Let $m, \phi \in M$. If m is an atom and $u_\alpha^M(m) \not\leq \phi$, then $m \leq \neg\Box_\alpha \phi$.*

Proof. *Assume m is an atom and $u_\alpha^M(m) \not\leq \phi$. Let $x \in M$. By proposition (3), the consequent is equivalent to the implication if $u_\alpha^M(x) \leq \phi$, then $m \wedge x = \bot$. Assume $u_\alpha^M(x) \leq \phi$. Since m is an atom, the infimum of m and x is the least element provided that $m \not\leq x$. For the purpose of reaching a contradiction, assume that $m \leq x$. Since $u_\alpha^M(-)$ preserves arbitrary joins, it is monotonic. Therefore, $u_\alpha^M(m) \leq u_\alpha^M(x)$. By assumption that $u_\alpha^M(m) \not\leq \phi$, we conclude that $u_\alpha^M(x) \not\leq \phi$. However, we assumed that $u_\alpha^M(x) \leq \phi$. We reached a contradiction. Therefore, $m \not\leq x$ proving the claim.*

Untestable Properties in the Kahr-Moore-Wang Class

Charles Jordan[⋆] and Thomas Zeugmann[⋆⋆]

Division of Computer Science,
Hokkaido University, N-14, W-9, Sapporo 060-0814, Japan
{skip,thomas}@ist.hokudai.ac.jp

Abstract. Property testing is a kind of randomized approximation in which one takes a small, random sample of a structure and wishes to determine whether the structure satisfies some property or is far from satisfying the property. We focus on the testability of classes of first-order expressible properties, and in particular, on the classification of prefix-vocabulary classes for testability. The main result is the untestability of $[\forall\exists\forall, (0, 1)]_=$. This is a well-known class and minimal for untestability. We discuss what is currently known about the classification for testability and briefly compare it to other classifications.

Keywords: property testing, logic, randomized algorithms.

1 Introduction

In property testing, we take a random sample of some large structure and wish to distinguish between the case that it has some desired property and the case that it is far from having the property. We focus on the testability of first-order expressible properties, and in particular on the classification of prefix-vocabulary classes of first-order logic for testability.

Rubinfeld and Sudan [21] and Blum *et al.* [4] introduced the notion of property testing in the context of formal verification. The basic idea was soon extended by Goldreich *et al.* [11], who represented graphs as binary functions and focused on testing graph properties. We omit a detailed history of testing, see the recent introduction to testing graph properties by Goldreich [10], two recent surveys by Ron [19, 20], and older surveys by Fischer [7] and Ron [18].

We are particularly interested in the testability of properties expressible in subclasses of first-order logic, and review relevant work in Subsection 1.1.

Here, we show that there exist untestable graph properties expressible with quantifier prefix $\forall\exists\forall$ when equality is allowed (see Section 3 for a formal statement). Taking into account the related work described in Subsection 1.1 and using the notation of Definition 7, the current classification for testability is the following.

[⋆] Supported by a Grant-in-Aid for JSPS Fellows under Grant No. 2100195209.

[⋆⋆] Supported by MEXT Grant-in-Aid for Scientific Research on Priority Areas under Grant No. 21013001.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 176–186, 2011.

- Testable classes
  1. Monadic first-order logic: $[all, (\omega)]_=$.
  2. Ackermann's class with equality: $[\exists^*\forall\exists^*, all]_=$.
  3. Ramsey's class: $[\exists^*\forall^*, all]_=$.
- Untestable classes
  1. $[\forall^3\exists, (0, 1)]_=$.
  2. $[\forall\exists\forall, (0, 1)]_=$.

We are especially interested in determining the testability of variants of the Gödel class (i.e., classes whose prefix contains at least $\forall^2\exists$) as this would suffice to complete the classification for the special case of predicate logic with equality. The above classification is consistent with several other well-known classifications, such as that for the finite model property (see, e.g., Chapter 6 of Börger *et al.* [5], for docility (or finite satisfiability, see Kolaitis and Vardi [16]) and for associated 0-1 laws for fragments of existential second-order logic (see Kolaitis and Vardi [16]). It would be interesting to know if the classification for testability coincides with one of these classifications.

The rest of the paper is organized as follows. First, we review related work in Subsection 1.1. Definitions and notation are in Section 2. The main result is presented in Section 3.

1.1 Related Work

Alon *et al.* [3] proved that the regular languages are testable, implying that monadic first-order is testable given the well-known results of Büchi [6] or McNaughton and Papert [17]. Alon *et al.* [2] were the first to directly consider the classification problem for testability, restricted to properties of undirected, loop-free graphs. They proved the testability of all such properties expressible by prenex sentences with quantifier prefix $\exists^*\forall^*$, and also showed that there exists an untestable property expressible with quantifier prefix $\forall^*\exists^*$ (examining the proof shows that $\forall^{12}\exists^5$ suffices).

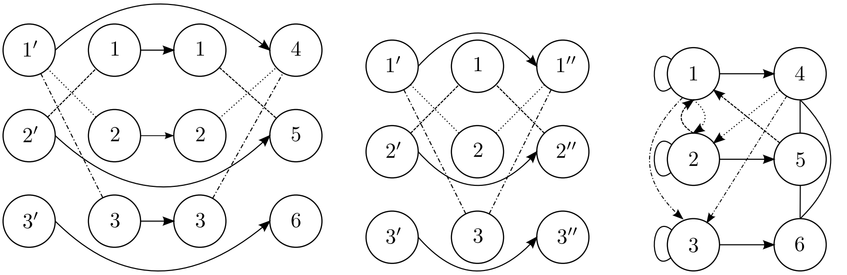
Both of these are (restrictions of) well-known classes. Jordan and Zeugmann extended [13] the positive result to the full Ramsey's class ($[\exists^*\forall^*, all]_=$), proved [12] the testability of Ackermann's class with equality ($[\exists^*\forall\exists^*, all]_=$), and sharpened [14] the negative result to prefixes $\forall^3\exists$, $\forall^2\exists\forall$, $\forall\exists\forall\exists$ and $\forall\exists\forall^2$ (on directed graphs with equality). This paper sharpens these last three prefixes and proves that $[\forall\exists\forall, (0, 1)]_=$ is a minimal prefix class for untestability. This particular class is the restriction of the Kahr-Moore-Wang [15] class (plus equality) to directed graphs.

It is easy to show that $[\forall\exists\forall, (0, 1)]$ (even without equality) has infinity axioms[1]. Vedø [22] showed that a 0-1 law does not hold for second-order existential logic when the first-order part is in this class (again, even without equality).

The current paper sharpens some (prefixes $\forall^2\exists\forall$, $\forall\exists\forall\exists$, $\forall\exists\forall^2$) of the results of Jordan and Zeugmann [14] and so we briefly outline the improvement that allows us to minimize the prefix considered. The untestable property considered in [14]

[1] An infinity axiom is a sentence that has only infinite models.

is closely related to the untestable property of Alon *et al.* [2], but modified to minimize the number of quantifiers used. These properties are essentially first-order expressible versions of checking an explicitly given isomorphism between two graphs[2]. In fact, restricting the properties to checking an explicitly given isomorphism between undirected, bipartite graphs (see Figure 1(a)) maintains hardness for testing. However, graph isomorphism seems to require one to discuss at least four vertices simultaneously (because one wishes to state that an edge is present iff its image is present and the edges are disjoint in general).



(a) A graph with P_b (b) A graph with P_e (c) A graph with P_f

Fig. 1. Properties P_b, P_e and P_f

Sharing one of the partitions (see Figure 1(b)) would seem to remove the need for four quantifiers. The resulting property is perhaps closer to a variant of *function* isomorphism, e.g., for functions $f, g\colon [n] \to \{0,1\}^n$ where the bit i of $f(j)$ is 1 if there is an edge from j in the leftmost partition to i in the middle partition and likewise for $g(j)$ and the right partition. This property is not first-order expressible, but there is a somewhat tedious first-order encoding that is similar (see Figure 1(c) and the formula in Section 3).

The connection with function isomorphism allows us to leverage recent work on the testability of (Boolean) function isomorphism and use recent ideas and techniques from Alon and Blais [1] to prove Lemma 1.

2 Preliminaries

The goal in property testing is always to distinguish structures that have some property from those that are far from having the property. Here, we focus on first-order expressible properties of directed graphs and so we begin with the necessary definitions.

Definition 1. *A graph is an ordered pair $G = (V, E)$, where V is a finite set and $E \subseteq V \times V$ a binary relation defined on V.*

We generally identify V with the first n naturals $[n] := \{1, \ldots, n\}$ and call $\#(G) := |V| = n$ the size of a graph G. Let $\mathcal{G}^n$ be the set of graphs of size n and

[2] Graph isomorphism is generally hard for testing, see, e.g., Fischer and Matsliah [8].

$\mathcal{G} := \cup_{n \geq 0} \mathcal{G}^n$ be the set of all (finite) graphs. Note that our graphs are directed and may contain loops.

A property $P \subseteq \mathcal{G}$ of graphs is any set of graphs. We are particularly interested in first-order expressible properties. Our logic is a basic first-order predicate logic with equality. There are no function or constant symbols. We focus on first-order properties of graphs, and so the only predicate symbol (besides the special symbol $=$) is the binary edge symbol E.

A sentence φ defines a property in the natural way,

$$P_\varphi := \{G \mid G \in \mathcal{G}, G \models \varphi\} .$$

We require a distance between graphs and properties, which we define in the following way. We denote the symmetric difference of sets by $\triangle$ and let E^A and E^B be the edge predicates of A and B, respectively.

Definition 2. *Let $A = (V, E^A)$ and $B = (V, E^B)$ be two graphs defined such that $|V| = n$. The distance between A and B is*

$$\mathrm{dist}(A, B) := |E^A \triangle E^B|/n^2 .$$

The distance generalizes to properties in the obvious way, $\mathrm{dist}(A, P) := \min_{B \in P} \mathrm{dist}(A, B)$. Definition 2 results in a typical model of testing based on the dense graph model introduced by Goldreich *et al.* [11]. We now proceed to the remaining testing definitions.

Definition 3. *An ε-*tester *for property P is a randomized algorithm that makes queries for the existence of edges in a graph A. The tester must accept with probability at least $2/3$ if A has P and must reject with probability at least $2/3$ if $\mathrm{dist}(A, P) \geq \varepsilon$.*

Definition 4. *Property P is called* testable *if there is some function $c(\varepsilon)$ and for every $\varepsilon > 0$, an ε-tester for P such that the tester makes at most $c(\varepsilon)$ queries.*

Note that the query complexity is bounded by a function that does not depend on the size of the graphs. We allow different ε-testers for each $\varepsilon > 0$ and so this is a non-uniform model. However, we are focused on proving untestability and our results hold even in the non-uniform case.

Next, we will define *indistinguishability*, a relation on properties introduced by Alon *et al.* [2] that preserves testability. However, testers can focus on *loops* and distinguish between structures that have an asymptotically small difference (because the number of loops is asymptotically dominated by the number of non-loops). We therefore begin with an alternative definition of distance (using this in place of Definition 2 makes testing (strictly) more difficult, but our result holds even when we use Definition 2). In the following, $\oplus$ denotes exclusive-or.

Definition 5. *Let $n \in \mathbb{N}$ and let U be any universe such that $|U| = n$. Furthermore, let $A = (U, E^A)$ and $B = (U, E^B)$ be any two graphs with universe U. For notational convenience, let*

$$d_1(A, B) := \frac{|\{x \mid x \in U \text{ and } E^A(x, x) \oplus E^B(x, x)\}|}{n} , \text{ and}$$

$$d_2(A,B) := \frac{|\{(x_1,x_2) \mid x_1,x_2 \in U,\ x_1 \neq x_2,\ \textit{and } E^A(x_1,x_2) \oplus E^B(x_1,x_2)\}|}{n(n-1)} .$$

The mr-distance *between A and B is*

$$\mathrm{mrdist}(A,B) := \max\{d_1(A,B), d_2(A,B)\} .$$

Definition 6. *Two properties P and Q of graphs are* indistinguishable *if they are closed under isomorphisms and for every $\varepsilon > 0$ there exists an N_ε such that for any graph A with universe of size $n \geq N_\varepsilon$, if A has P then* $\mathrm{mrdist}(A,Q) \leq \varepsilon$ *and if A has Q then* $\mathrm{mrdist}(A,P) \leq \varepsilon$*.*

An important property of indistinguishability is that it preserves testability. The proof of the following is analogous to that given in Alon *et al.* [2].

Theorem 1. *If P and Q are indistinguishable, then P is testable if and only if Q is testable.*

In fact, as the proof constructs an ε-tester for P by iterating an $\varepsilon/2$-tester for Q three times, one can also relate the query complexities of P and Q. Many proofs of hardness for testability rely on Yao's Principle [23], an interpretation of von Neumann's minimax theorem for randomized computation. For completeness, we state the version that we use.

Principle 1 (Yao's Principle). *If there is an $\varepsilon \in (0,1)$ and a distribution over $\mathcal{G}^n$ such that all deterministic testers with complexity c have an error-rate greater than $1/3$ for property P, then property P is not testable with complexity c.*

The definition of "testable" is of course our usual one involving random testers. In general, one seeks to show that for sufficiently large n and some increasing function $c := c(n)$, there is a distribution of inputs such that all deterministic testers with complexity c have error-rates greater than $1/3$.

Finally, we briefly define the notation we use to specify prefix-vocabulary classes. See Börger *et al.* [5] for details and related material.

Definition 7. *Let Π be a string over the four-character alphabet $\{\exists, \forall, \exists^*, \forall^*\}$. Then $[\Pi, (0,1)]_=$ is the set of sentences in prenex normal form which satisfy the following conditions.*

1. *The quantifier prefix is contained in the regular language given by Π (for technical reasons, one usually treats $\exists$ and $\forall$ as matching the relevant quantifier and also the empty string).*
2. *There are zero* (0) *monadic predicate symbols.*
3. *In addition to the equality predicate* (=)*, there is at most one* (1) *binary predicate symbol.*
4. *There are no other predicate symbols.*

That is, $[\Pi, (0,1)]_=$ is the set of prenex sentences in the logic defined above whose quantifier prefixes match Π. If the second component of the specification is *all*, then conditions two and three are removed (any number of predicate symbols with any arities are acceptable).

3 An Untestable Property

Our goal in this section is Theorem 2.

Theorem 2. *The prefix class* $[\forall\exists\forall, (0,1)]_=$ *is not testable.*

We begin by outlining the proof. First, we define P_f, a property expressible in the class $[\forall\exists\forall, (0,1)]_=$ which, as described in Subsection 1.1, is in some sense a somewhat tedious but first-order expressible variant of checking (explicit) isomorphism of undirected bipartite graphs in tripartite graphs. We then define a variant P_2, in which the isomorphism is not explicitly given and we must test whether there exists some suitable isomorphism. Although this increases the complexity of deciding the problem from checking an isomorphism to finding one, it does not change hardness for testing. We show that P_2 and P_f are indistinguishable and so P_2 is testable iff P_f is testable. Finally, we prove directly that P_2 is untestable, even with $o(\sqrt{n})$ queries, using an argument based on a recent proof by Alon and Blais [1].

Proof (Theorem 2). We begin by defining P_f. Formally, it is the set of graphs satisfying the following conjunction of four clauses (see Figure 1(c) for an example).

$$\begin{aligned}
\forall x \exists y \forall z: \quad & \{ ((\neg E(x,x) \wedge \neg E(z,z) \wedge x \neq z) \rightarrow E(x,z)) \\
\wedge \quad & (E(x,x) \rightarrow (E(x,y) \wedge \neg E(y,y) \wedge [(\neg E(z,z) \wedge E(x,z)) \rightarrow y = z])) \\
\wedge \quad & (\neg E(x,x) \rightarrow (E(y,x) \wedge E(y,y) \wedge [(E(z,z) \wedge E(z,x)) \rightarrow y = z])) \\
\wedge \quad & ((E(x,x) \wedge E(z,z)) \rightarrow [\neg E(y,y) \wedge E(x,y) \wedge (E(x,z) \leftrightarrow E(y,z))]) \}
\end{aligned}$$

A graph satisfies this formula if the following conditions are all satisfied.

1. The nodes without loops form a complete subgraph.
2. For every node x with a loop, there is exactly one y without a loop such that there is an edge from x to y.
3. For every node y without a loop, there is exactly one x with a loop such that there is an edge from x to y.
4. For all nodes x, z with loops, and y the unique node without a loop such that $E(x,y)$, it holds that $E(x,z)$ iff $E(y,z)$.

Property P_2 below is similar to P_f, except that the isomorphism is not explicitly given.

Definition 8. *A graph* $G = (V, E)$ *has* P_2 *if it satisfies the following conditions.*

1. *There is a partition*[3] $V_1, V_2 \subseteq V$ *such that* $|V_1| = |V_2|$, *there are loops* $(E(x,x))$ *on all* $x \in V_1$ *and no loops* $(\neg E(x,x))$ *for all* $x \in V_2$.
2. *The nodes without loops form a complete subgraph.*
3. *There are no edges from a node with a loop to a node without a loop.*

[3] V_1, V_2 partition V if $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$.

4. *There exists a bijection* $b\colon V_1 \to V_2$ *such that if* x, z *have loops, then* $E(x, z)$ *iff* $E(b(x), z)$.

It is not difficult to show that properties P_f and P_2 are indistinguishable.

Claim 1. *Properties* P_f *and* P_2 *are indistinguishable.*

Proof (Claim 1). Let $\varepsilon > 0$ be arbitrary and let $N_\varepsilon = \varepsilon^{-1}$. Assume that G has property P_2 and that $\#(G) > N_\varepsilon$. We will show that $\mathrm{mrdist}(G, P_f) < \varepsilon$.

Graph G has P_2 and so there is a bijection satisfying Condition 4 of Definition 8. We therefore add the edges $E(i, b(i))$ making the isomorphism (from V_1 to V_2) explicit. The resulting graph G_f has P_f.

We have made exactly $n/2$ modifications, all to non-loops, and $n - 1 \geq N_\varepsilon$, so $\mathrm{mrdist}(G, P_f) \leq \mathrm{mrdist}(G, G_f) = 1/2(n - 1) < \varepsilon$.

The converse is analogous; given a G that has P_f, simply remove the $n/2$ edges from loops to non-loops after using them to construct a suitable bijection b. □ Claim 1

Properties P_f and P_2 are indistinguishable and so (by Theorem 1), it suffices to show that P_2 is is untestable. Lemma 1 below is stronger than necessary, and actually implies a $\Omega(\sqrt{n})$ lower bound for testing P_f per the discussion following Theorem 1. □ Theorem 2

Lemma 1. *Fix* $0 < \varepsilon < 1/2$. *Any* ε*-tester for* P_2 *must perform* $\Omega(\sqrt{n})$ *queries.*

Proof (Lemma 1). The proof is via Yao's Principle (cf. Principle 1), and so we define two distributions, D_{no} and D_{yes} and show that all deterministic testers have an error-rate greater than 1/3 for property P_2 when the input is chosen randomly from D_{no} with probability 1/2 and from D_{yes} with probability 1/2.

In the following, we consider a distribution over graphs of sufficiently large size $2n$, and an arbitrary fixed partition of the vertices into V_1 and V_2 such that $|V_1| = |V_2| = n$ (for example, let the vertices be the integers $V := [2n]$, $V_1 := [n]$ and $V_2 := V \backslash V_1$).

We begin with D_{no}, defined as the following distribution.

1. Place a loop on each vertex in V_1 and place no loops in V_2.
2. Place each possible edge (except loops) in $V_1 \times V_1$ and $V_2 \times V_1$ uniformly and independently with probability 1/2.

That is, D_{no} is the uniform distribution of graphs (with this particular partition) satisfying the first three conditions of P_2.

Next, we define D_{yes} as the following.

1. Choose uniformly a random bijection $\pi\colon V_1 \to V_2$.
2. Place a loop on each vertex in V_1 and place no loops in V_2.
3. For each possible edge $(i, j \neq i) \in V_1 \times V_1$, uniformly and independently place *both* (i, j) and $(\pi(i), j)$ with probability 1/2 (otherwise place *neither*).

It is easy to see that D_{yes} generates only positive instances. Next, we show that D_{no} generates negative instances with high probability.

Lemma 2. *Fix $0 < \varepsilon < 1/2$ and let n be sufficiently large. Then,*

$$\Pr_{G \sim D_{no}} [\operatorname{dist}(G, P_2) \leq \varepsilon] = o(1)\,.$$

Proof (Lemma 2). D_{no} is the uniform distribution over graphs of size $2n$ with a particular partition satisfying the first three conditions of P_2. Let G_ε be the set of graphs G' of size $2n$ satisfying these conditions and such that $\operatorname{dist}(G', P_2) \leq \varepsilon$ (regardless of partition).

Counting the number of such graphs shows

$$|G_\varepsilon| \;\leq\; \binom{2n}{n} 2^{n(n-1)} n! \sum_{i=0}^{\lceil \varepsilon 2n^2 \rceil} \binom{2n^2}{i} \;\leq\; \binom{2n}{n} 2^{n(n-1)} n! 2^{H(\epsilon)2n^2}\,,$$

where $H(\varepsilon) := -\varepsilon \log \varepsilon - (1-\varepsilon)\log(1-\varepsilon)$ is the binary entropy function (cf. Lemma 16.19 in Flum and Grohe [9] for the bound on the summation).

Distribution D_{no} produces each of $2^{n(n-1)}2^{n^2}$ graphs with equal probability, and so

$$\begin{aligned}\Pr_{G \sim D_{\text{no}}} [\operatorname{dist}(G, P_2) \leq \varepsilon] \;&\leq\; \frac{|G_\varepsilon|}{2^{n(n-1)+n^2}} \;\leq\; \binom{2n}{n} n! 2^{H(\epsilon)2n^2} / 2^{n^2} \\ &\approx\; \frac{4^n n! 2^{H(\epsilon)2n^2}}{\sqrt{\pi n} 2^{n^2}} \;=\; o(1)\,.\end{aligned}$$

The approximation is asymptotically tight, which suffices. □ Lemma 2

We have shown that D_{yes} generates only positive instances and that (with high probability) D_{no} generates instances that are ε-far from P_2. Next, we show that (again, with high probability) the two distributions look the same to testers making only $o(\sqrt{n})$ queries.

The proof is similar to a proof by Alon and Blais [1]. We begin by defining two random processes, P_{no} and P_{yes}, which answer queries from testers and generate instances according to D_{no} and D_{yes}, respectively.

Process P_{no} is defined in the following way.

1. Choose uniformly a random bijection $\pi\colon V_1 \to V_2$.
2. Intercept all queries from the tester and respond as follows.
   (a) To queries $E(i, i)$ with $i \in V_1$: respond 1.
   (b) To queries $E(i, i)$ with $i \in V_2$: respond 0.
   (c) To queries $E(i, j)$ with $i \in V_1$ and $j \in V_2$: respond 0.
   (d) To queries $E(i, j)$ with $i \neq j \in V_1$: quit if we have queried $E(\pi(i), j)$, otherwise respond 1 or 0 randomly with probability 1/2 in each case.
   (e) To queries $E(i, j)$ with $i \in V_2$ and $j \in V_1$: quit if we have queried $E(\pi^{-1}(i), j)$, otherwise respond 1 or 0 randomly with probability 1/2 in each case.
3. When the process has quit or the tester has finished its queries, complete the generated instance in the following way. First, fix the edges that were queried according to our answer. Next, place loops on each vertex in V_1, no loops in V_2 and no edges from V_1 to V_2. Place each remaining possible edge, place it (uniformly, independently) with probability 1/2, *ignoring* π.

We define P_{yes} in the same way, except for the final step. When P_{yes} quits or the tester finishes, it fixes the edges that were queried according to its answers, and *also* fixes the corresponding edges (when relevant) according to π. More precisely, for each fixed $E(i,j)$ with $i \neq j \in V_1$, we also fix $E(\pi(i),j)$ and for fixed $E(i,j)$ with $i \in V_2, j \in V_1$, we also fix $E(\pi^{-1}(i),j)$, in both cases the same as our response to $E(i,j)$ (not randomly). The remaining edges are placed as in P_{no}.

Note that P_{no} generates instances according to D_{no} and P_{yes} generates instances according to D_{yes}. In addition, P_{yes} and P_{no} behave identically until they quit or answer all queries. In particular, if a tester does not cause the process to quit, the distribution of responses of its queries is identical for the two processes. We show that, with high probability, a tester that makes $o(\sqrt{n})$ queries does not cause either process to quit.

Lemma 3. *Let T be a deterministic tester which makes $o(\sqrt{n})$ queries, and let T interact with P_{yes} or P_{no}. In both cases,*

$$\Pr\left[T \text{ causes the process to quit}\right] = o(1)\,.$$

Proof (Lemma 3). The condition causing the process to quit is identical in P_{yes} and P_{no}. The probability that any pair of queries $E(i,j)$ and $E(i',j')$ cause the process to quit is at most

$$\Pr\left[i' = \pi(i) \text{ or } i = \pi(i')\right] \leq \frac{(n-1)!}{n!} = 1/n\,.$$

The tester makes at most $o(\sqrt{n})$ queries and so

$$\Pr\left[T \text{ causes the process to quit}\right] \leq o(\sqrt{n})^2 O(1/n) = o(1)\,.$$

□ Lemma 3

Any deterministic tester T which makes $o(\sqrt{n})$ queries can only distinguish between D_{yes} and D_{no} with probability $o(1)$, but it must accept D_{yes} with probability $2/3$, and reject D_{no} with probability $2/3 - o(1)$. It is impossible for T to satisfy both conditions, and the lemma follows from Principle 1. □ Lemma 1

4 Conclusion

Property testing is a kind of randomized approximation, where we take a small, random sample of a structure and seek to determine whether the structure has a desired property or is far from having the property. We focused on the classification problem for testability, wherein we seek to determine exactly which prefix vocabulary classes are testable and which are not. The main result of this paper is the untestability of $[\forall\exists\forall,(0,1)]_{=}$, a sharpening of the results of [14]. This class is a minimal class for untestability.

As mentioned in Subsection 1.1, the current classification for testability closely resembles several other classifications (e.g., those for the finite model property, docility and associated second-order 0-1 laws) and it would be interesting to determine whether it coincides with one of these. In particular, determining the testability of variants of the Gödel class would complete the classification for the special case of predicate logic with equality.

References

[1] Alon, N., Blais, E.: Testing Boolean function isomorphism. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 394–405. Springer, Heidelberg (2010)

[2] Alon, N., Fischer, E., Krivelevich, M., Szegedy, M.: Efficient testing of large graphs. Combinatorica 20(4), 451–476 (2000)

[3] Alon, N., Krivelevich, M., Newman, I., Szegedy, M.: Regular languages are testable with a constant number of queries. SIAM J. Comput. 30(6), 1842–1862 (2001)

[4] Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. J. of Comput. Syst. Sci. 47(3), 549–595 (1993)

[5] Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Springer, Heidelberg (1997)

[6] Büchi, J.R.: Weak second-order arithmetic and finite-automata. Z. Math. Logik Grundlagen Math. 6, 66–92 (1960)

[7] Fischer, E.: The art of uninformed decisions. Bulletin of the European Association for Theoretical Computer Science 75, 97–126 (2001)

[8] Fischer, E., Matsliah, A.: Testing graph isomorphism. SIAM J. Comput. 38(1), 207–225 (2008)

[9] Flum, J., Grohe, M.: Parametrized Complexity Theory. Springer, Heidelberg (2006)

[10] Goldreich, O.: Introduction to testing graph properties. Technical Report TR10-082, Electronic Colloquium on Computational Complexity (ECCC) (May 2010)

[11] Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. J. ACM 45(4), 653–750 (1998)

[12] Jordan, C., Zeugmann, T.: Relational properties expressible with one universal quantifier are testable. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 141–155. Springer, Heidelberg (2009)

[13] Jordan, C., Zeugmann, T.: A note on the testability of Ramsey's class. In: Kratochvíl, J., Li, A., Fiala, J., Kolman, P. (eds.) TAMC 2010. LNCS, vol. 6108, pp. 296–307. Springer, Heidelberg (2010)

[14] Jordan, C., Zeugmann, T.: Untestable properties expressible with four first-order quantifiers. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.) LATA 2010. LNCS, vol. 6031, pp. 333–343. Springer, Heidelberg (2010)

[15] Kahr, A.S., Moore, E.F., Wang, H.: Entscheidungsproblem reduced to the $\forall\exists\forall$ case. Proc. Nat. Acad. Sci. U.S.A. 48, 365–377 (1962)

[16] Kolaitis, P.G., Vardi, M.Y.: 0-1 laws for fragments of existential second-order logic: A survey. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 84–98. Springer, Heidelberg (2000)

[17] McNaughton, R., Papert, S.: Counter-Free Automata. M.I.T. Press, Cambridge (1971)

[18] Ron, D.: Property testing. In: Rajasekaran, S., Pardalos, P.M., Reif, J.H., Rolim, J. (eds.) Handbook of Randomized Computing, vol. II, pp. 597–649. Kluwer Academic Publishers, Dordrecht (2001)

[19] Ron, D.: Property testing: A learning theory perspective. Found. Trends Mach. Learn. 1(3), 307–402 (2008)

[20] Ron, D.: Algorithmic and analysis techniques in property testing. Found. Trends Theor. Comput. Sci. 5(2), 73–205 (2009)

[21] Rubinfeld, R., Sudan, M.: Robust characterizations of polynomials with applications to program testing. SIAM J. Comput. 25(2), 252–271 (1996)

[22] Vedø, A.: Asymptotic probabilities for second-order existential Kahr-Moore-Wang sentences. J. Symbolic Logic 62(1), 304–319 (1997)

[23] Yao, A.C.C.: Probabilistic computations: Toward a unified measure of complexity. In: 18th Annual Symposium on Foundations of Computer Science, pp. 222–227. IEEE Computer Society, Los Alamitos (1977)

Characterizing Definability of Second-Order Generalized Quantifiers[⋆]

Juha Kontinen[1] and Jakub Szymanik[2]

[1] Department of Mathematics and Statistics, University of Helsinki
juha.kontinen@helsinki.fi
[2] Institute of Artificial Intelligence, University of Groningen
jakub.szymanik@gmail.com

Abstract. We study definability of second-order generalized quantifiers. We show that the question whether a second-order generalized quantifier $\mathcal{Q}_1$ is definable in terms of another quantifier $\mathcal{Q}_2$, the base logic being monadic second-order logic, reduces to the question if a quantifier $\mathcal{Q}_1^\star$ is definable in $\mathrm{FO}(\mathcal{Q}_2^\star, <, +, \times)$ for certain first-order quantifiers $\mathcal{Q}_1^\star$ and $\mathcal{Q}_2^\star$. We use our characterization to show new definability and non-definability results for second-order generalized quantifiers. In particular, we show that the monadic second-order majority quantifier Most^1 is not definable in second-order logic.

1 Introduction

The notion of generalized quantifier goes back to Mostowski [1] and Lindström [2]. Generalized quantifiers were first mainly studied in the framework of model theory. The study of generalized quantifiers extended to the context of finite model theory via applications to descriptive complexity theory. We refer to [3] and [4] for surveys of first-order generalized quantifiers in finite model theory. Generalized quantifiers have been also extensively studied in the formal semantics of natural language (see [5] for a survey).

The study of second-order generalized quantifiers is a relatively new and unexplored area in finite model theory. On the other hand, second-order logic (SO) and its many fragments have been studied extensively starting from Fagin's characterization of NP in terms of existential second-order logic [6]. Second-order generalized quantifiers were first studied in the context of finite structures by Burtschick and Vollmer [7]. Shortly after, Andersson [8] studied the expressive power of families of second-order generalized quantifiers determined by the syntactic types of quantifiers. In [9,10,11] Kontinen studied definability questions of second-order generalized quantifiers. In the case of first-order quantifiers, definability of a quantifier Q in a logic $\mathcal{L}$ means that the class of structures, used to interpret Q, is axiomatizable in $\mathcal{L}$. In the second-order case, the analogous

[⋆] The first author was supported by grant 127661 of the Academy of Finland. The second author was supported by NWO Vici grant 277-80-001.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 187–200, 2011.

concept of definability was formulated in [9,10]. In this article, we give a computationally motivated characterization for the notion of definability of second-order generalized quantifiers.

Burtschick and Vollmer [7] noticed that second-order generalized quantifiers can be used to logically characterize complexity classes defined in terms of so-called *Leaf Languages*. The leaf languages approach in computational complexity theory, introduced by Bovet, Crescenzi, and Silvestri [12], is a unifying approach to define complexity classes. The central idea behind this approach is to generalize the conditions under which, e.g., a Turing machine or an automaton accepts its input. Many complexity classes can be defined in this context in terms of suitable leaf languages. On the other hand, a complexity class defined in terms of a leaf language B can be under certain conditions characterized logically in terms of a logic of the form:

$$\mathcal{Q}_B\,\mathrm{FO},$$

where $\mathcal{Q}_B$ is a second-order generalized quantifier corresponding to the language B. In the context of leaf languages, polynomial time non-deterministic Turing machines can be sometimes replaced by non-deterministic finite automata (so-called finite leaf automata) without a significant increase in complexity [13]. Galota and Vollmer [14] showed that complexity classes defined in terms of finite leaf automata can be logically characterized in terms of monadic second-order generalized quantifiers. This result nicely extends the well known [15,16,17] characterization of regular languages in terms of monadic second-order logic (MSO).

The definability theory of second-order generalized quantifiers has some similarities and differences compared to that of first-order generalized quantifiers. For example, it was observed in [9] that the binary second-order existential quantifier cannot be defined in terms of any monadic second-order generalized quantifiers. This result is in contrast with fact (a corollary of a result of Andersson [8]) that all classes of finite first-order structures are already definable in terms of monadic second-order generalized quantifiers. In this paper we prove a general result characterizing the question when a quantifier $\mathcal{Q}$ is definable in $\mathrm{MSO}(\mathcal{Q}', +)$, where $+$ denotes the built-in addition relation. We assume the built-in addition in order to unleash the expressive power embodied by MSO. Recall that, while MSO corresponds to regular languages over strings, $\mathrm{MSO}(+)$ corresponds to the linear fragment of the polynomial hierarchy (LINH) on strings [18]. It is possible to formulate our characterization also in the case where the base logic is full second-order logic instead of $\mathrm{MSO}(+)$.

Our characterization is based on a logical formalization of an idea of Torán [19]. Torán studied oracle separations in the counting hierarchy and noticed that there is essentially no difference between an oracle Turing machine writing an oracle query on its query tape and a logarithmic time Turing machine writing an address on its random access tape. He used this analogy to show that an oracle separation result for classes in the polynomial counting hierarchy implies a real separation for the corresponding classes in the logarithmic counting hierarchy LINCH (equivalently in DLOGTIME-uniform TC^0). We show that a second-order generalized quantifier $\mathcal{Q}_1$ is definable in the logic $\mathrm{MSO}(\mathcal{Q}_2, +)$ iff for certain first-order encodings $\mathcal{Q}_i^\star$ of $\mathcal{Q}_i$, $\mathcal{Q}_1^\star$ is definable in $\mathrm{FO}(\mathcal{Q}_2^\star, +, \times)$. It is worth noting that the latter condition implies that $\mathcal{Q}_1^\star$ is AC^0 (Turing) reducible to $\mathcal{Q}_2^\star$. We use our characterization to show new definability and non-definability results for second-order generalized quantifiers. In particular, we show that the monadic second-order majority quantifier Most^1 is not definable in second-order logic. This aswers the question left open in [20] (see also [21]), where second-order generalized quantifiers were used to model collective quantification in natural language. For the sake of brevity, we do not discuss the role and use of quantifiers in formal semantics in this paper.

2 Preliminaries

In this article all structures are assumed to be finite. The universe of a structure $\mathfrak{A}$ is denoted by A. Without loss of generality, we may assume that A is always of the form $\{0, \dots, m\}$ for some $m \in \mathbb{N}$. For a logic $\mathcal{L}$, the set of τ-formulas of $\mathcal{L}$ is denoted by $\mathcal{L}[\tau]$. If ϕ is a τ-sentence, then the class of τ-models of ϕ is denoted by $\mathrm{Mod}(\phi)$. A class K of τ-models is said to be axiomatizable in a logic $\mathcal{L}$, if $K = \mathrm{Mod}(\phi)$ for some sentence $\phi \in \mathcal{L}[\tau]$. For logics $\mathcal{L}$ and $\mathcal{L}'$, we write $\mathcal{L} \leq \mathcal{L}'$, if for every τ and every sentence $\phi \in \mathcal{L}[\tau]$ there is a sentence $\psi \in \mathcal{L}'[\tau]$ such that $\mathrm{Mod}(\phi) = \mathrm{Mod}(\psi)$. The set of natural numbers is denoted by $\mathbb{N}$ and $\mathbb{N}^*$ denotes the set $\mathbb{N} \setminus \{0\}$.

Sometimes we assume that our structures (and logics) are equipped with auxiliary built-in relations. In addition to the built-in ordering $<$, which is interpreted naturally, we also use the ternary relations $+$ and $\times$. The relations $+$ and $\times$ are defined as

$$
\begin{aligned}
+(i, j, k) &\Leftrightarrow i + j = k, \\
\times(i, j, k) &\Leftrightarrow i \times j = k.
\end{aligned}
$$

The relation BIT is a further important relation which is defined by: $\mathrm{BIT}(a, j)$ holds iff the bit of order 2^j is 1 in the binary representation $\mathrm{bin}(a)$ of a. The presence of built-in relations is signalled, e.g., by the notation $\mathrm{FO}(<)$. It is well known that $\mathrm{FO}(<, +, \times) \equiv \mathrm{FO}(<, \mathrm{BIT})$ (see [22]). Note that $<$ is easily definable in $\mathrm{FO}(+)$ and hence, in the precence of $+$, we sometimes do not mention $<$ explicitely.

We assume that the reader is familiar with the basics of computational complexity theory. Below, we recall certain results from descriptive complexity theory. It is instructive to note that many of the logics considered in this article correspond to interesting complexity classes. We mention first the logic $\mathrm{FO}(<, +, \times)$ which corresponds exactly to the so-called logarithmic hierarchy (LH). This class is the logarithmic analogue of the polynomial hierarchy (PH), corresponding to SO [23], defined in terms of alternating Turing machines (ATM) running in polynomial time with $O(1)$ alternations. In between LH and PH we have the linear hierarchy (LINH) corresponding to the logic $\mathrm{MSO}(+)$ over strings [18].

In this article also majority quantifiers are dicussed and studied. It is well-known that majority quantifiers can be used to logically characterize counting computations. The following counting hierarchies are relevant for this article: the logarithmic counting hierarchy (LCH), the linear counting hierarchy (LINCH), and the (polynomial) counting hierarchy (CH) all of which can be defined, with analogous resource bounds as LH, LINH, and PH, in terms of so-called Threshold Turing machines [24]. On the logical side, majority quantifiers (defined in Section 2.1) can be used to provide logical counterparts for these classes: $\mathrm{FO}(\mathrm{M},+,\times) \equiv$ LCH [25], $\mathrm{FO}(\mathrm{Most}^1,<) \equiv$ LINCH (over strings) [26], and $\mathrm{FO}(\mathrm{Most}^k)_{k\in\mathbb{N}^*} \equiv$ CH [27]. Furthermore, in circuit complexity, it is known that LH corresponds exactly to DLOGTIME-uniform AC^0 and LCH to DLOGTIME-uniform TC^0 [25]. Also, DLOGTIME-uniform $\mathrm{AC}^0[p]$ (AC^0 with unbounded fan-in MOD_p gates) corresponds on the logical side to $\mathrm{FO}(\mathrm{D}_k,+,\times)$ [25].

2.1 Generalized Quantifiers

In this section we briefly recall some basics of generalized quantifiers.

Let $\tau = \{P_1,\dots,P_r\}$ be a relational vocabulary, where P_i is l_i-ary for $1 \leq i \leq r$, and Q a class of τ-structures closed under isomorphisms. The class Q gives rise to a generalized quantifier which we also denote by Q. The tuple $s = (l_1,\dots,l_r)$ is the *type* of the quantifier Q.

Definition 1. *The extension* FO(Q) *of first-order logic by a quantifier* Q *is defined as follows:*

1. *The formula formation rules of* FO *are extended by the rule: if for* $1 \leq i \leq r$, $\phi_i(\overline{x}_i)$ *is a formula and* $\overline{x}_i$ *is an* l_i*-tuple of pairwise distinct variables then* $Q\overline{x}_1,\dots,\overline{x}_r\,(\phi_1(\overline{x}_1),\dots,\phi_r(\overline{x}_r))$ *is a formula.*
2. *The satisfaction relation of* FO *is extended by the rule:*

$$\mathfrak{A} \models Q\overline{x}_1,\dots,\overline{x}_r\,(\phi_1(\overline{x}_1),\dots,\phi_r(\overline{x}_r)) \textit{ iff } (A,\phi_1^{\mathfrak{A}},\dots,\phi_r^{\mathfrak{A}}) \in Q,$$

where $\phi_i^{\mathfrak{A}} = \{\overline{a} \in A^{l_i} \mid \mathfrak{A} \models \phi_i(\overline{a})\}$.

We say that a quantifier Q is definable in a logic $\mathcal{L}$ if the class Q is axiomatizable in $\mathcal{L}$. Note that Q is trivially definable in FO(Q). If $\mathcal{L}$ has the substitution property and is closed under FO-operations, then definability of Q in $\mathcal{L}$ implies that $\mathrm{FO}(Q) \leq \mathcal{L}$. So, among such logics, FO(Q) is the minimal logic in which Q is definable.

Example 1. The following quantifiers will be discussed in the following sections. Suppose $S \subseteq \mathbb{N}$ and $k \in \mathbb{N}$.

$$\begin{aligned}\exists &= \{(A,P) \mid P \subseteq A \text{ and } P \neq \emptyset\}\\ \mathrm{M} &= \{(A,P) \mid P \subseteq A \text{ and } |P| > |A|/2\}\\ Q_S &= \{(A,P) \mid P \subseteq A \text{ and } |P| \in S\}\end{aligned}$$

If S is of the form $\{kn \mid n \in \mathbb{N}\}$ for some $k \in \mathbb{N}$, we denote Q_S by D_k.

We will also refer to the *vectorizations* of the quantifiers D_k and M later. The nth vectorization of D_k is the following quantifier

$$\mathrm{D}_k^n = \{(A, P) \mid P \subseteq A^n \text{ and } |P| = 0 \bmod k\},$$

and the nth vectorization of M is

$$\mathrm{M}^n = \{(A, P) \mid P \subseteq A^n \text{ and } |P| > |A^n|/2\}.$$

Let us then turn to second-order generalized quantifiers. Let $t = (s_1, \ldots, s_w)$, where $s_i = (l_1^i, \ldots, l_{r_i}^i)$ is a tuple of positive integers for $1 \le i \le w$. A second-order structure of type t is a structure of the form $(A, P_1, \ldots, P_w)$, where $P_i \subseteq \mathcal{P}(A^{l_1^i}) \times \cdots \times \mathcal{P}(A^{l_{r_i}^i})$.

Definition 2. *A second-order generalized quantifier $\mathcal{Q}$ of type t is a class of structures of type t such that $\mathcal{Q}$ is closed under isomorphisms.*

A quantifier $\mathcal{Q}$ is *monadic* if $l_j^i = 1$ for all $1 \le j \le r_i$ and $1 \le i \le w$. Let us look at some examples of second-order generalized quantifiers.

Example 2. Suppose $S \subseteq \mathbb{N}$ and $k \in \mathbb{N}$.

$$\begin{aligned}
\exists_k^2 &= \{(A, P) \mid P \subseteq \mathcal{P}(A^k) \text{ and } P \neq \emptyset\} \\
\text{Even} &= \{(A, P) \mid P \subseteq \mathcal{P}(A) \text{ and } |P| \text{ is even}\} \\
\text{Even}' &= \{(A, P) \mid P \subseteq \mathcal{P}(A) \text{ and } \forall X \in P\ (|X| \text{ is even})\} \\
\text{Most}^k &= \{(A, P) \mid P \subseteq \mathcal{P}(A^k) \text{ and } |P| > 2^{|A|^k - 1}\} \\
\mathcal{Q}_S &= \{(A, P) \mid P \subseteq \mathcal{P}(A) \text{ and } |P| \in S\}
\end{aligned}$$

Analogously to the first-order case, if S is of the form $\{kn \mid n \in \mathbb{N}\}$ for some $k \in \mathbb{N}$, we denote $\mathcal{Q}_S$ by $\mathcal{D}_k$.

The first example is the familiar k-ary second-order existential quantifier. The quantifier Even says that a formula holds for an even number of subsets of the universe. On the other hand, the quantifier Even′ says that all the subsets satisfying a formula have an even cardinality. The quantifier Most^k is the k-ary second-order version of M expressing that a formula holds for more than half of the k-ary relations.

Definition 3. *The extension* FO($\mathcal{Q}$) *of* FO *by a quantifier $\mathcal{Q}$ is defined as follows:*

1. *The formula formation rules of* FO *are extended by the rule: if for $1 \le i \le w$, $\phi_i(\overline{X}_i)$ is a formula and $\overline{X}_i = (X_{1,i}, \ldots, X_{r_i,i})$ is a tuple of pairwise distinct predicate variables such that the arity of $X_{j,i}$ is l_j^i for $1 \le j \le r_i$, then*

$$\mathcal{Q}\overline{X}_1, \ldots, \overline{X}_w\, (\phi_1(\overline{X}_1), \ldots, \phi_w(\overline{X}_w))$$

   is a formula.
2. *Satisfaction relation of* FO *is extended by the rule:*

$$\mathfrak{A} \models \mathcal{Q}\overline{X}_1, \ldots, \overline{X}_w\, (\phi_1, \ldots, \phi_w) \text{ iff } (A, \phi_1^{\mathfrak{A}}, \ldots, \phi_w^{\mathfrak{A}}) \in \mathcal{Q},$$

   where $\phi_i^{\mathfrak{A}} = \{\overline{R} \in \mathcal{P}(A^{l_1^i}) \times \cdots \times \mathcal{P}(A^{l_{r_i}^i}) \mid \mathfrak{A} \models \phi_i(\overline{R})\}$.

2.2 Definability

Recall that a first-order generalized quantifier Q is definable in a logic $\mathcal{L}$ if the class Q is axiomatizable in $\mathcal{L}$. This condition can be reformulated as follows assuming $\mathcal{L}$ has the substitution property:

Proposition 1. *A first-order quantifier Q is definable in a logic $\mathcal{L}$ if and only if $\mathcal{L} \equiv \mathcal{L}(Q)$.*

How do we formalize definability for second-order quantifiers? Intuitively, e.g., the monadic second-order existential quantifier $\exists^2_1$ is definable in a logic $\mathcal{L}$ if there is a uniform way to express

$$\exists^2_1 X \psi(X)$$

for any formula $\psi(X)$ in the logic $\mathcal{L}$. Over a model $\mathfrak{A}$, $\psi(X)$ defines a collection of subsets

$$\{C \subseteq A \mid \mathfrak{A} \models \psi(C)\},$$

so the problem is to find a way to express the non-emptyness of this collection in a way which does not depend on the particular formula $\psi(X)$. This was formalized in [10] using second-order relations.

Definition 4. *Let $\mathcal{L}$ be a logic, $t = (s_1, \dots, s_w)$ a second-order type, and let $\mathcal{G}_1, \dots, \mathcal{G}_w$ be first-order quantifier symbols of types $s_1, \dots, s_w$.*

1. *The logic $\mathcal{L}(\mathcal{G}_1, \dots, \mathcal{G}_w)$ is obtained by extending the syntax of $\mathcal{L}$ in terms of the quantifiers $\mathcal{G}_1, \dots, \mathcal{G}_w$.*
2. *The models of $\mathcal{L}(\mathcal{G}_1, \dots, \mathcal{G}_w)$ are of the form $\mathcal{A} = (\mathfrak{A}, G_1, \dots, G_w)$, where $\mathfrak{A}$ is a first-order model and*

$$G_i \subseteq \mathcal{P}(A^{l^i_1}) \times \dots \times \mathcal{P}(A^{l^i_{r_i}}).$$

3. *The quantifiers $\mathcal{G}_i$ are interpreted using the relations G_i:*

$$\mathcal{A} \models \mathcal{G}_i \bar{x}_1, \dots, \bar{x}_{r_i}(\phi_1(\bar{x}_1), \dots, \phi_{r_i}(\bar{x}_{r_i}))$$

iff $(\phi_1^{\mathcal{A}}, \dots, \phi_{r_i}^{\mathcal{A}}) \in G_i$.

Note that if $\phi \in \mathcal{L}(\mathcal{G}_1, \dots, \mathcal{G}_w)$ is a sentence of vocabulary $\tau = \emptyset$. Then

$$\mathrm{Mod}(\phi) = \{(A, G_1, \dots, G_w) \mid (A, G_1, \dots, G_w) \models \phi\}$$

corresponds to a second-order generalized quantifier of type t. This observation can be used to formalize definability of second-order generalized quantifiers. Below, we assume that $\mathcal{L}$ is closed under substitution.

Definition 5. *Let $\mathcal{Q}$ be a quantifier of type t. The quantifier $\mathcal{Q}$ is definable in a logic $\mathcal{L}$ if there is $\phi \in \mathcal{L}(\mathcal{G}_1, \dots, \mathcal{G}_w)$ of vocabulary $\sigma = \emptyset$ such that for any t-structure $(A, G_1, \dots, G_w)$,*

$$(A, G_1, \dots, G_w) \models \phi \Leftrightarrow (A, G_1, \dots, G_w) \in \mathcal{Q}.$$

The following was shown in [10]:

Theorem 2. *If $\mathcal{Q}$ is definable in $\mathcal{L}$ then $\mathcal{L} \equiv \mathcal{L}(\mathcal{Q})$.*

The converse of Theorem 2 does not hold:

Theorem 3 ([10]). *There is a quantifier $\mathcal{Q}$ of type* $((1))$ *which is not definable in* FO *and satisfies* $\mathrm{FO} \equiv \mathrm{FO}(\mathcal{Q})$.

Definability questions of second-order quantifiers has been studied in [10,11,27]. We recall the following results.

Theorem 4 ([11]). *Let t be type and $\mathcal{B}_t$ the collection of all second-order quantifiers of types less than t. Then there is a quantifier $\mathcal{Q}$ of type t such that $\mathcal{Q}$ is not definable in* $\mathrm{SO}(\mathcal{B}_t)$.

Theorem 4 is proved with respect to a natural ordering of the types of second-order generalized quantifiers. Theorem 4 is existential in nature and does not give us a concrete non-definable quantifier. It was observed in [9] that it is not so difficult to find concrete quantifiers which cannot be defined using any monadic quantifiers. Denote by $\mathbb{Q}$ the collection of all monadic second-order generalized quantifiers.

Theorem 5 ([9]). *The quantifier $\exists^2_2$ is not definable in* $\mathrm{FO}(\mathbb{Q})$.

It is worth noting that the logic $\mathrm{FO}(\mathbb{Q})$ is capable of defining all classes of first-order structures (cf. Theorem 6.2 in [8]). Finally, we recall the following result about second-order majority quantifiers:

Theorem 6 ([27]). *The quantifier $\exists^2_k$ is definable in* $\mathrm{FO}(\mathrm{Most}^k)$.

It interesting to note that definability of Most^1 in the logic SO would imply that $\mathrm{PH} \equiv \mathrm{CH}$ in computational complexity. This observation was discussed in [20]. In this paper we show that the quantifier Most^1 is not definable in SO, but, analogously to Theorem 3, this non-definability result does not imply that $\mathrm{PH} \subsetneq \mathrm{CH}$.

3 Characterizing Definability

The computational analogue of a first-order generalized quantifier is the notion of an oracle (see [22]). Let Q be a quantifier of vocabulary τ and $\mathcal{L}$ a logic. The idea is that in $\mathcal{L}(Q)$ we can query "without a cost" if a definable τ-structure $\mathfrak{A}$ is a member of the class Q. Recall that a second-order generalized quantifier $\mathcal{Q}$ of type $((1))$ is definable, e.g., in SO if there is a sentence $\phi \in \mathrm{SO}(\mathcal{G})$ such that for all second-order structures (A, G):

$$(A, G) \models \phi \Leftrightarrow (A, G) \in \mathcal{Q}\,. \tag{1}$$

It is not immediately clear how to view this notion in computational terms. The set G corresponds to a local first-order quantifier and, if we treat G as an oracle, then in (1) we are infact trying to define a property oracles. One way to proceed is to formalize definability of a quantifier $\mathcal{Q}$ in terms of oracle Turing machines that treat (a suitable initial segment) the oracle as part of the input. However, in this article we do not follow that idea as there is a more familiar route to take. An important observation here is that the set G can be of exponential size compared to the domain A. This observation can be used to show that SO-definability of $\mathcal{Q}$ reduces to logarithmic time definability.

Our proof is based on a logical version of an idea of Torán [19] showing that there is essentially no difference between an oracle Turing machine writing an oracle query on its query tape, and a logarithmic time Turing machine writing an address on its random access tape. In other words, an oracle in the setting of polynomial time machines can be viewed as an input to a logarithmic time machine. We use a logical version of this idea: we show that SO and the relation G in (1) can be replaced by FO and a unary relation P by passing from A to a domain of cardinality $2^{|A|}$.

In this section we mainly restrict attention to monadic second-order generalized quantifiers. We interpret definability of quantifiers in logics with built-in relations in the natural way. For example, a second-order quantifier $\mathcal{Q}$ of type $((1))$ is definable in MSO(+) if there is $\phi \in \mathrm{MSO}(\mathcal{G}, +)$ such that for all structures $(A, +, G)$: $(A, +, G) \models \phi \Leftrightarrow (A, G) \in \mathcal{Q}$. In particular, Theorem 2 can be proved analogously in this setting.

Next we define a first-order encoding of a second-order structure of type t, for a monadic t. We use the fact that there is a one-to-one correspondence between integers $m \in B = \{0, \ldots, 2^n - 1\}$ and subsets of $A = \{0, \ldots, n-1\}$ seen as length-n binary numbers. Therefore, relations of A can be encoded in terms of tuples of elements of B and, further, sets of relations of A by relations of B.

Definition 6. *Let $t = (s_1, \ldots, s_w)$ be a type where $s_i = (1, \ldots, 1)$ is of length r_i for $1 \leq i \leq w$. Let $\mathfrak{A} = (A, G_1, \ldots, G_w)$ be a t-structure where $A = \{0, \ldots, n-1\}$ and $G_i \subseteq \mathcal{P}(A) \times \cdots \times \mathcal{P}(A)$. Denote by $\hat{\mathfrak{A}} = (B, P_1, \ldots, P_w)$ the following first-order structure of vocabulary $\tau = \{P_1, \ldots, P_w\}$, where P_i is a r_i-ary predicate, and*

1. $B = \{0, \ldots, 2^n - 1\}$,
2. $P_i = \{(j_1, \ldots, j_{r_i}) \in B^{r_i} \mid (J_1, \ldots, J_{r_i}) \in G_i\}$, *where, for* $1 \leq k \leq r_i$, $\mathrm{bin}(j_k)$ *is given by* $s_0 \cdots s_{n-1}$, *and* $s_l = 1 \Leftrightarrow l \in J_k$.

For a quantifier $\mathcal{Q}$ of type t, we denote by $\mathcal{Q}^\star$ the first-order quantifier of vocabulary τ defined by

$$\mathcal{Q}^\star := \{\hat{\mathfrak{A}} : \mathfrak{A} \in \mathcal{Q}\}.$$

Note that the quantifier $\mathcal{Q}^\star$ has only structures in cardinalities of the form 2^n and that $|G_i| = |P_i|$ for $1 \leq i \leq w$. We are now ready for the main result of this article (see the Appendix for the proof).

Theorem 7. *Let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be monadic quantifiers. Then $\mathcal{Q}_1$ is definable in* $\mathrm{MSO}(\mathcal{Q}_2, +)$ *if and only if $\mathcal{Q}_1^\star$ is definable in* $\mathrm{FO}(\mathcal{Q}_2^\star, +, \times)$.

Let us then discuss some corollaries of Theorem 7. We need the following definition.

Definition 7. *Let $t = (s_1, \ldots, s_w)$ and τ be as in Definition 6. Let $\mathcal{Q}$ be a quantifier of type t. The quantifier $\mathcal{Q}$ is* numerical *if there is a relation $T \subseteq \mathbb{N}^w$ such that for all t-structures $(A, P_1, \ldots, P_w)$*

$$(A, P_1, \ldots, P_w) \in \mathcal{Q} \Leftrightarrow (|P_1|, \ldots, |P_w|) \in T.$$

We denote $\mathcal{Q}$ by $\mathcal{Q}_T$ and by Q_T the first-order numerical quantifier (defined analogously) of vocabulary τ.

It is easy to see that, for a numerical $\mathcal{Q}_T$, the quantifier $\mathcal{Q}_T^\star$ (see Definition 6) is just the restriction of the corresponding first-order quantifier Q_T to the cardinalities 2^n:

$$\mathcal{Q}_T^\star = \{(A, P_1, \ldots, P_w) \in Q_T : |A| = 2^n \text{ for some } n \in \mathbb{N}\}.$$

This observation allows us to show the following (see the Appendix for the proof):

Theorem 8. *Let $\mathcal{Q}_T$ be a numerical quantifier and $k \in \mathbb{N}$. Then*

1. *$\mathcal{Q}_T$ is definable in* MSO(+) *iff Q_T is definable in* $\mathrm{FO}(+, \times)$.
2. *$\mathcal{Q}_T$ is definable in* $\mathrm{MSO}(\mathcal{D}_k, +)$ *iff Q_T is definable in* $\mathrm{FO}(\mathrm{D}_k, +, \times)$.
3. *$\mathcal{Q}_T$ is definable in* $\mathrm{MSO}(\mathrm{Most}^1, +)$ *iff Q_T is definable in* $\mathrm{FO}(\mathrm{M}, +, \times)$.

The following lemma can be now used.

Lemma 1. *Let $S \subseteq \mathbb{N}$, p a prime, and $q > 1$ relatively prime to p. Then*

1. *Q_S is definable in* $\mathrm{FO}(+, \times)$ *iff S either finite or cofinite.*
2. D_q *is not definable in* $\mathrm{FO}(\mathrm{D}_p, +, \times)$.

Proof. The first claim follows from non-definability of the language PARITY in $\mathrm{FO}(+, \times)$ [28,29] (see Theorem 4.3 in [30]). The second claim goes back to [31].

By combining Theorem 8 and Lemma 1 we can show the following.

Corollary 1. *Let $S \subseteq \mathbb{N}$, p a prime, and $q > 1$ relatively prime to p. Then*

1. *$\mathcal{Q}_S$ is definable in* MSO(+) *iff S is either finite or cofinite.*
2. *$\mathcal{D}_q$ is not definable in* $\mathrm{MSO}(\mathcal{D}_p, +)$.

It is possible to replace MSO(+) by SO in Theorem 7. The idea is that, if $\mathcal{Q}_1$ is definable in $\mathrm{SO}(\mathcal{Q}_2)$, then in the defining formula, for some k, only relations of arity at most k are quantified. We do not present this generalization in detail in this article but only consider the special case of the quantifier Most^1.

Theorem 9. *The quantifier* Most^1 *is not definable in* SO.

Proof. It suffices to show that Most^1 is not definable in $\mathrm{FO}(\exists^2_k)$ for any k. Note also that $(\mathrm{Most}^1)^\star$ is the restriction of M to the cardinalities 2^n.

An analogous translation as in Theorem 7 can be used to show that definability of Most^1 in SO implies that, for some k, the quantifier M is definable in $\mathrm{FO}(+,\times)$ over cardinalities 2^{n^k}. Over these cardinalities, we could then express PARITY in the logic $\mathrm{FO}(+,\times)$. This contradicts the result of [28,29].

In order to translate the quantifier $\exists^2_k$ to the logic $\mathrm{FO}(+,\times)$, we redefine the structure $\hat{\mathfrak{A}}$ (see Definition 6) to have a domain of the form $\{0,\ldots,2^{n^k}-1\}$. Now we can use the fact that there is a one-to-one correspondence between integers $m \in \{0,\ldots,2^{n^k}-1\}$ and k-ary relations R of $\{0,\ldots,n-1\}$. In other words, by using the lexicographic ordering on k-tuples, a relation R can be encoded by a binary string of length n^k corresponding to the binary representation of a unique integer $m < 2^{n^k}$. It is straightforward to adjust the translation in the proof of Theorem 7 to this setting.

4 Conclusion

We have shown that definability of second-order generalized quantifiers can be reduced to definability of first-order generalized quantifiers. We have indicated a couple of corollaries to our characterization but surely there is more to be done here. Also, as discussed in connection to Theorem 9, it is possible to replace MSO in terms of SO and to prove a result analogous to Theorem 7 in this case. In particular, Theorem 9 solves the open problem proposed in [20], where we studied the collective meanings of natural language quantifiers. It suggests, as we argued in [20], that the type-shifting strategy [32] to define the meanings of natural language quantification might be too restricted in its computational power. It is likely that second-order logic is not enough to capture natural language semantics. Another interpretation would be that everyday language does not realize hard collective quantifiers (for sure they are marginal at best) due to their complexity.

References

1. Mostowski, A.: On a generalization of quantifiers. Fund. Math. 44, 12–36 (1957)
2. Lindström, P.: First order predicate logic with generalized quantifiers. Theoria 32, 186–195 (1966)
3. Väänänen, J.: Generalized quantifiers, an introduction. In: Väänänen, J. (ed.) ESSLLI 1997. LNCS, vol. 1754, pp. 1–17. Springer, Heidelberg (2000)
4. Ebbinghaus, H.D., Flum, J.: Finite model theory. In: Perspectives in Mathematical Logic, 2nd edn. Springer, Heidelberg (1999)
5. Peters, S., Westerståhl, D.: Quantifiers in Language and Logic. Clarendon Press, Oxford (2006)
6. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Complexity of Computation (Proc. SIAM-AMS Sympos. Appl. Math., New York, 1973). SIAM–AMS Proc., vol. VII, pp. 43–73. Amer. Math. Soc., Providence (1974)

7. Burtschick, H.J., Vollmer, H.: Lindström quantifiers and leaf language definability. Int. J. Found. Comput. Sci. 9(3), 277–294 (1998)
8. Andersson, A.: On second-order generalized quantifiers and finite structures. Ann. Pure Appl. Logic 115(1-3), 1–32 (2002)
9. Kontinen, J.: Definability of second order generalized quantifiers. PhD thesis, University of Helsinki (2005)
10. Kontinen, J.: Definability of second order generalized quantifiers. Arch. Math. Logic 49(3), 379–398 (2010)
11. Kontinen, J.: The hierarchy theorem for second order generalized quantifiers. J. Symbolic Logic 71(1), 188–202 (2006)
12. Bovet, D.P., Crescenzi, P., Silvestri, R.: A uniform approach to define complexity classes. Theor. Comput. Sci. 104(2), 263–283 (1992)
13. Peichl, T., Vollmer, H.: Finite automata with generalized acceptance criteria. Discrete Mathematics and Theoretical Computer Science 4, 179–192 (2001)
14. Galota, M., Vollmer, H.: A generalization of the Büchi-Elgot-Trakhtenbrot theorem. In: Fribourg, L. (ed.) CSL 2001 and EACSL 2001. LNCS, vol. 2142, pp. 355–368. Springer, Heidelberg (2001)
15. Büchi, J.R., Elgot, C.C.: Decision problems of weak second order arithmetics and finite automata, Part I. Notices of the American Mathematical Society 5, 834 (1958)
16. Büchi, J.R.: On a decision method in restricted second-order arithmetic. In: Proceedings Logic, Methodology and Philosophy of Sciences 1960. Stanford University Press, Stanford (1962)
17. Trakhtenbrot, B.A.: Finite automata and logic of monadic predicates. Doklady Akademii Nauk SSSR 140, 326–329 (1961) (in Russian)
18. More, M., Olive, F.: Rudimentary languages and second-order logic. Math. Logic Quart. 43(3), 419–426 (1997)
19. Torán, J.: Structural properties of the counting hierarchies. PhD thesis, Facultat d'Informatica de Barcelona, Barcelona, Spain (1988)
20. Kontinen, J., Szymanik, J.: A remark on collective quantification. Journal of Logic, Language and Information 17(2), 131–140 (2008)
21. Szymanik, J.: Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language. PhD thesis, Universiteit van Amsterdam (2009)
22. Immerman, N.: Descriptive complexity. In: Graduate Texts in Computer Science. Springer, New York (1999)
23. Stockmeyer, L.J.: The polynomial-time hierarchy. Theor. Comput. Sci. 3(1), 1–22 (1977)
24. Parberry, I., Schnitger, G.: Parallel computation with threshold functions. J. Comput. System Sci. 36(3), 278–302 (1988)
25. Barrington, D.A.M., Immerman, N., Straubing, H.: On uniformity within NC^1. J. Comput. System Sci. 41(3), 274–306 (1990)
26. Kontinen, J., Niemistö, H.: Extensions of MSO and the monadic counting hierarchy. Information and Computation 209(1), 1–19 (2011)
27. Kontinen, J.: A logical characterization of the counting hierarchy. ACM Trans. Comput. Log. 10(1) (2009)
28. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Systems Theory 17(1), 13–27 (1984)
29. Ajtai, M.: Σ^1_1-formulae on finite structures. Ann. Pure Appl. Logic 24(1), 1–48 (1983)

30. Barrington, D.A.M., Immerman, N., Lautemann, C., Schweikardt, N., Thérien, D.: First-order expressibility of languages with neutral letters or: The Crane Beach conjecture. J. Comput. System Sci. 70(2), 101–127 (2005)
31. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Proc. 19th Annual ACM Symposium on Theory of Computing, pp. 77–82. ACM Press, New York (1987)
32. Winter, Y.: Flexibility principles in Boolean semantics. The MIT Press, London (2001)
33. Hesse, W., Allender, E., Barrington, D.A.M.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comput. System Sci. 65(4), 695–716 (2002); Special issue on complexity (2001) (Chicago, IL)

Appendix

Proof (of Theorem 7)

To simplify notation, we assume that the type of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ is $((1,1))$ and $((1),(1))$, respectively.

Let us first assume that $\mathcal{Q}_1$ is definable in the logic $\mathrm{MSO}(\mathcal{Q}_2,+)$. Then there is a sentence $\phi \in \mathrm{MSO}(\mathcal{Q}_2,\mathcal{G},+)$ such that for all structures $(A,+,G)$

$$(A,+,G) \models \phi \Leftrightarrow (A,G) \in \mathcal{Q}_1 .$$

We shall next show that there is a sentence $\phi^* \in \mathrm{FO}(\mathcal{Q}_2^{\star}+,\times)[\{P\}]$, where P is binary, such that for all structures $\mathfrak{A} = (A,G)$:

$$(A,+,G) \models \phi \Leftrightarrow (B,P,<,+,\times) \models \phi^*, \tag{2}$$

where $(B,P) = \hat{\mathfrak{A}}$ (see Definition 6). We define ϕ^* via the following translation:

$$\begin{aligned}
x_i = x_j &\rightsquigarrow x_i = x_j\\
x_i + x_j = x_k &\rightsquigarrow x_i + x_j = x_k\\
Y_i(x_j) &\rightsquigarrow \mathrm{BIT}(y_i, n-(x_j+1))\\
\mathcal{G}x_i, x_j(\psi_1(x_i),\psi_2(x_j)) &\rightsquigarrow \exists z_1 \exists z_2 \Big(P(z_1,z_2) \wedge \bigwedge_{1\le i\le 2} \forall (w<n)(\psi_i^*(w)\\
&\qquad\qquad \leftrightarrow \mathrm{BIT}(z_i, n-(w+1)))\Big)\\
\psi \wedge \theta &\rightsquigarrow \psi^* \wedge \theta^*\\
\neg\psi &\rightsquigarrow \neg\psi^*\\
\exists x_i \psi &\rightsquigarrow \exists x_i (x_i < n \wedge \psi^*(x_i))\\
\exists Y_i \psi &\rightsquigarrow \exists y_i \psi^*\\
\mathcal{Q}_2\, Y_i, Y_j(\psi(Y_i),\theta(Y_j)) &\rightsquigarrow \mathcal{Q}_2^{\star}\, y_i, y_j(\psi^*(y_i),\theta^*(y_j))
\end{aligned}$$

It is now straigthforward to show that for all formulas $\psi \in \mathrm{MSO}(\mathcal{Q}_2,\mathcal{G},+)$, structures (A,G), and assignments s

$$(A,+,G) \models_s \psi \Leftrightarrow (B,P,<,+,\times) \models_{s^*} \psi^*,$$

where the assignment s^* is defined such that $s^*(x_i) = s(x_i)$ for all first-order variables x_i, and, if $s(Y_i) = D \subseteq \{0, \dots, n-1\}$, then $s^*(y_i)$ is the unique $d < 2^n$ whose binary representation is given by $s_0 \cdots s_{n-1}$ where $s_j = 1 \iff j \in D$.

In the formula translation, we use the predicate BIT, which is FO$(+, \times)$-definable, to recover the set D from the integer d. By the above translation, the sentence

$$\exists n(|B| = 2^n \wedge \phi^*)$$

of the logic FO$(\mathcal{Q}_2^\star, +, \times)$ now defines the quantifier $\mathcal{Q}_1^\star$.

Let us then show the converse implication. Assume that $\phi \in$ FO$(\mathcal{Q}_2^\star, +, \times)$ defines the quantifier $\mathcal{Q}_1^\star$. The idea is now to translate $\phi \in$ FO$(\mathcal{Q}_2^\star, +, \times)$ to $\phi' \in$ MSO$(\mathcal{Q}_2, \mathcal{G}, +)$ such that for all $\mathfrak{A} = (A, G)$:

$$(A, +, G) \models \phi' \Leftrightarrow (B, P, <, +, \times) \models \phi. \tag{3}$$

Analogously to the first translation, we encode integers in the domain $B = \{0, \dots, 2^n - 1\}$ in terms of subsets $X \subseteq \{0, \dots, n-1\}$. We use the following formulas $X = Y$, $X < Y$, $X + Y = Z$, and $X \times Y = Z$ expressing arithmetic operations on binary numbers. The first three formulas are FO$(+)$-expressible, and the fourth is expressible in the logic FO$(\mathrm{M}, +, \times) \leq$ MSO$(+)$ [33]. The translation $\phi \rightsquigarrow \phi'$ is now defined as follows.

$$\begin{aligned}
P(x_i, x_j) &\rightsquigarrow \mathcal{G} z_1, z_2(X_i(z_1), X_j(z_2)) \\
x_i = x_j &\rightsquigarrow X_i = X_j \\
x_i < x_j &\rightsquigarrow X_i < X_j \\
x_i + x_j = x_k &\rightsquigarrow X_i + X_j = X_k \\
x_i \times x_j = x_k &\rightsquigarrow X_i \times X_j = X_k \\
\psi \wedge \phi &\rightsquigarrow \psi' \wedge \phi' \\
\neg\psi &\rightsquigarrow \neg\psi' \\
\exists x_i \psi(x_i) &\rightsquigarrow \exists X_i \psi'(X_i) \\
Q_2^\star x_i, x_j(\psi(x_i), \theta(x_j)) &\rightsquigarrow \mathcal{Q}_2\, X_i, X_j(\psi'(X_i), \theta'(X_j))
\end{aligned}$$

It is straightforward to show that this translation works as intended. In particular, it follows that the sentence $\phi' \in$ MSO$(\mathcal{Q}_2, \mathcal{G}, +)$ now defines the quantifier $\mathcal{Q}_1$.

Proof (of Theorem 8) The proof is based on the fact that each of the logics FO$(<, +, \times)$, FO$(\mathrm{D}_k, +, \times)$, and FO$(\mathrm{M}, +, \times)$ is closed under logical reductions. Suppose that $\mathcal{Q}_T$ is of type $t = (s_1, \dots, s_w)$ and let τ denote the vocabulary of the corresponding first-order quantifier Q_T (see Definition 6).

Let us consider claim 2. By Theorem 7 it suffices to show that the following are equivalent:

(a) $\mathcal{Q}_T^\star$ is definable in FO$(\mathcal{D}_k^\star, +, \times)$
(b) Q_T is definable in FO$(\mathrm{D}_k, +, \times)$

Recall that the quantifiers $\mathcal{Q}_T^\star$ and $\mathcal{D}_k^\star$ are the restrictions of the quantifiers Q_T and D_k to cardinalities of the form 2^n, respectively. Let us first note that (a) is equivalent with

(c) $\mathcal{Q}_T^\star$ is definable in $\mathrm{FO}(\mathrm{D}_k, +, \times)$.

First of all, since $\mathcal{D}_k^\star$ is easily definable in $\mathrm{FO}(\mathrm{D}_k, +, \times)$ using the $\mathrm{FO}(+, \times)$-expressible predicate $x = 2^y$, it follows that (a) $\Rightarrow$ (c). Assume then that (c) holds and let $\phi \in \mathrm{FO}(\mathrm{D}_k, +, \times)$ define $\mathcal{Q}_T^\star$. Define a sentence ψ as follows:

$$\psi := \exists n(|A| = 2^n \wedge \phi(\mathrm{D}_k \,/\, \mathcal{D}_k^\star)).$$

Since the quantifier $\mathcal{Q}_T^\star$ contains structures only in cardinalities of the form 2^n it is easy to see that $\psi \in \mathrm{FO}(\mathcal{D}_k^\star, +, \times)$ also defines $\mathcal{Q}_T^\star$.

It now suffices to show that (b) and (c) are equivalent. Note first that (b) $\Rightarrow$ (c) can be easily proved using the predicate $x = 2^y$. We will show $(c) \Rightarrow$ (b). Here we use the fact that the logic $\mathrm{FO}(\mathrm{D}_k, +, \times)$ is closed under logical reductions. We will define Q_T (over all cardinalities) with the help of the quantifier $\mathcal{Q}_T^\star$. Let $\mathfrak{A}$ be a structure. If $|A| = 2^n$ for some $n \in \mathbb{N}$, then $\mathfrak{A} \in Q_T$ can be expressed in terms of the quantifier $\mathcal{Q}_T^\star$. Note that even if $|A|$ is not a power of two, it holds that the least m such that $|A| \leq 2^m$ satisfies $2^m \leq |A|^2$.

We will now sketch how the quantifier Q_T can be defined in terms of $\mathcal{Q}_T^\star$. Assume $\phi \in \mathrm{FO}(\mathrm{D}_k, +, \times)$ is a sentence defining $\mathcal{Q}_T^\star$. Let $\mathfrak{A} = (A, P_1, \ldots, P_w)$ be a τ-structure, where $A = \{0, \ldots, n-1\}$. We use the following facts:

1. There is a $\mathrm{FO}(<, +, \times)$-definable query I that maps $\mathfrak{A}$ to the structure $I(\mathfrak{A})$ which is isomorphic to

$$(\{0, \ldots, 2^m - 1\}, P_1, \ldots, P_w, <, +, \times),$$

where 2^m is the least power of two satisfying $n \leq 2^m$.

2. There is a sentence $\psi \in \mathrm{FO}(\mathrm{D}_k, +, \times)$ such that for all $\mathfrak{A}$:

$$\mathfrak{A} \models \psi \Leftrightarrow I(\mathfrak{A}) \models \phi.$$

Since Q_T is numerical, the sentence ψ now defines Q_T. The query I is easily definable in $\mathrm{FO}(<, +, \times)$; the domain of $I(\mathfrak{A})$ is defined as $\{(i,j) \in A^2 \mid in + j < 2^m\}$ (see [22] for more on first-order queries). The sentence ψ is constructed inductively (see e.g., Section 3.2 [22]) using, in particular, the fact that the second vectorization D_k^2 of D_k can be expressed in $\mathrm{FO}(\mathrm{D}_k, +, \times)$.

The claims 1 are 3 are proved analogously. For claim 3 we use the facts that $(\mathrm{Most}^1)^\star$ is the restriction of M to the cardinalities 2^n and that the second vectorization M^2 of M is definable in $\mathrm{FO}(\mathrm{M}, +, \times)$ (see [25]).

Countable Version of Omega-Rule

Grigori Mints

Stanford University,
Department of Philosophy
http://philosophy.stanford.edu/profile/Grigori+Mints/

Abstract. Omega-rule used by W. Buchholz to give an ordinal-free proof-theoretic analysis of Π^1_1-comprehension axiom has uncountable set of premises. We show how to make this set countable preserving the results and ideas of cut-elimination proof by Buchholz. The price is introduction of non-well founded derivation-like figures and use of continuous cut-elimination.

Keywords: cut-elimination, Π^1_1-analysis, Omega-rule.

1 Introduction

Ω-rule was introduced by W. Buchholz (cf. [2]) to give a proof-theoretic analysis of Π^1_1-comprehension axiom. This rule has uncountable set of premises. We show how to make this set countable preserving the results and ideas of proof by W. Buchholz. He defines a translation $d \to d^\infty$ from a familiar formal system of Π^1_1-analysis (with the axiom of mathematical induction and a rule

$$\frac{\Gamma, A(F)}{\Gamma, \exists X A(X)}$$

with suitable restrictions) into an infinitary system with the ω-rule for the first-order quantifier $\forall x$ and Ω-rule (see below) for the second order quantifier $\exists X$. Derivations in this infinitary system are defined inductively in a familiar way as well-founded trees proceeding from axioms by inference rules. Cut-elimination for the infinitary system is established by transfinite induction on derivations without use of proof-theoretic ordinals or Girard's computability predicates. Using this W. Buchholz developed in [5] an independent approach to Howard's ordinal $\phi_{\epsilon_{\Omega+1}}(0)$. One can try to use this feature to extend an approach to proof-theoretic ordinals from [1].

Cut-elimination procedure for the systems with Ω-rule consists of two stages. First, cuts over formulas properly containing the most complicated quantifiers $\exists X A(X)$ are eliminated in a standard way. This process generates some residual cuts formalized as $\tilde{\Omega}$-rules (see below) which still have uncountable number of premises. The second stage called collapsing eliminates $\tilde{\Omega}$-rules (i.e., remaining cuts) from the derivations of arithmetical formulas beginning from the uppermost "cuts". The whole uncountable tree ending in a $\tilde{\Omega}$-rule is reduced to just one

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 201–209, 2011.

of its subtrees. The result is a cut-free derivation of an arithmetical sequent by familiar rules including ω-rule, hence it is countable.

A countable version proposed in the present paper replaces uncountable branching in $\varOmega$, $\tilde{\varOmega}$-rules by the countable branching over all closed arithmetical sequents. The role of a “derivation” of an underivable sequent $\varGamma$ is taken by a familiar object, a proof-search tree $\mathcal{T}_\varGamma$ of $\varGamma$. This change makes the trees considered here countable but non-well-founded. For example a proof-search tree for a formula $\phi :\equiv \exists x(x = x \& x < 0)$ looks as follows:

$$\cfrac{\cfrac{0=0,\phi \qquad \cfrac{\cfrac{0<0,1=1,\phi \quad \overset{\vdots}{0<0,1<0,\phi}}{0<0,1=1\&1<0,\phi}}{0<0,\phi}}{0=0\&0<0,\phi}}{\exists x(x=x\&x<0)} \tag{1}$$

This figure is only “local” correct: each step is made by an inference rule and the leaf nodes are axioms.

However it turns out that both stages of cut-elimination still go through without fundamental changes if continuous cut-elimination (introduced by the author in [8]) is applied. This version of the standard cut-elimination transformations works with non-well-founded proof-figures like (1) from the bottom-up: it begins with the endsequent. Neccesary properties of such transformations (for example that one cut-reduction step preserves local correctness) cannot be proved by transfinite induction TI from the top (i.e., axioms) down, since TI is not applicable. They are proved by the bottom-up induction using the way in which any finite bottom part of the result of a transformation is constructed from sufficiently large chunks of the arguments. Details are developed in [8]. Even more detailed treatment easily formalizable in primitive recursive arithmetic and using finite notation for infinite derivations is given in [3]. These notations were applied in [4] to derive Takeuti’s reductions for finite derivations from infinitary cut-elimination with $\varOmega$-rule. The countable version presented below preserves geometric view of cut-elimination without breaking it into mosaic of finite reductions.

The price is introduction of non-well founded derivation-like figures and use of continuous cut-elimination. Probably some of this can be stated in the language of co-recursion, but we avoid that framework.

In fact we consider here only a subsystem treated in [4]. Unexplained notation is understood exactly as in [4]. In particular we restrict second order arithmetical formulas in the same way:

$\forall XA, \exists XA$ *is a formula only if* A *contains no second order quantifier and no second order variable except* X. Negation $\neg A$ is defined by classical de Morgan rules.

Notation

$$\begin{array}{c} q : \varGamma' \\ \hline\hline d_q : \varGamma \end{array}$$

indicates that some operation transforms given derivation q of Γ' into a derivation d_q of Γ. Notation

$$\frac{\Gamma' \quad \Gamma''}{\Gamma}\ \mathcal{O}$$

indicates that operation $\mathcal{O}$ transforms a pair of derivations of Γ and Γ' into a derivation of Γ.

Ω-rule from [4] can be pictured as follows:

$$\cfrac{\cfrac{q:\ \Delta, A(X)}{d_q:\ \ldots \Gamma, \Delta, \neg\forall X A(X) \ldots}}{\Gamma, \neg\forall X A(X)}\ \Omega \tag{2}$$

with a separate premise d_q for each cut-free derivation q of an arithmerical sequent $\Delta, A(X)$. This set of premises is uncountable since there are uncountably many derivations of arithmetical sequents. We choose one of these derivations in a canonical way, which makes the number of premises countable.

In more traditional notation this would be

$$\frac{\ldots \Gamma, \Delta, \neg\forall X A(X) \ldots}{\Gamma, \neg\forall X A(X)}\ \Omega$$

Precise definition in [4] looks as follows. Let Var_1 be the set of second order variables. For each formula $P := \forall X A(X)$ and derivation $d : \Gamma(d)$ of a sequent $\Gamma(d)$ let

$$P[X] := A(X),\ \Delta^P_{d,X} := \Gamma(d) - \{P[X]\}$$

$$|P| := \{(d, X) \in BI_0^\infty \times Var_1 : \Gamma(d) \text{ arithmetical } \& X \notin FV(\Delta^P_{d,X})\}$$

Now the rule is:

$$\frac{\ldots \Delta^P_q \ldots (q \in |P|)}{\neg P}\ \Omega_{\neg P}$$

where Y is an eigenvariable.

2 Proof Search Tree for Arithmetic

Definition 1. *A sequent is* arithmetical *if it does not contain set quantifiers (but may contain free set variables $X, Y, \ldots$). Following [4] we denote by $\mathcal{A}$ the set of all arithmetical sequents.*

A system BI_0^∞ in [4] is a familiar complete cut-free Tait-style system for first order arithmetic with ω-rule for numerical quantifier $\forall x$. The rules for proof search below are obviously instances of the rules of BI_0^∞.

Recall that a *proof-search tree* $\mathcal{T}_\Gamma$ of a sequent Γ according to some set of rules is a primitive recursive tree with Γ as the endsequent (root) where every potentially possible rule is eventually applied. We need it for the case of arithmetic where it can be described by the following inference rules.

Derived objects are *sequents*: finite sequences of arithmetical formulas without free first order variables. The empty sequence is interpreted as a constant $\bot$, i.e. false. We give only rules needed to derive prenex formulas. Let TRUE stand for the set of true primitive recursive closed formulas.

Inference rules:

$$\frac{}{R, \Gamma}\,Ax,\ \text{if } R \in \mathsf{TRUE} \text{ or } \Gamma \supset \{A, \neg A\}$$

$$\frac{\Gamma}{R, \Gamma}\,\mathbf{F},\ \text{if } R \in \mathsf{FALSE}$$

$$\frac{F[n], \Gamma, [n+1]\exists x F[x]}{[n]\exists x F[x], \Gamma}\,\exists$$

$$\frac{\ldots\ F[n], \Gamma\ \ldots\quad \text{all } n}{\forall x F[x], \Gamma}\,\forall$$

Let's describe the proof-search tree $\mathcal{T}_S$ for the sequent S by these rules. $\mathcal{T}_S$ is treated here as a primitive recursive function assigning to any node a belonging to the tree of finite sequences of natural numbers the symbol 0 (if a $\notin \mathcal{T}_S$) or a sequent S_{a}. In particular for the root $\emptyset$ of the tree $S_\emptyset = S$.

Any non-empty sequent S_{a} can be conclusion of exactly one rule to be denoted ρ_{a}. We place the l-th premise of the rule ρ_{a} it into the node $\mathrm{a} * l$.

The relation of the conclusion S_{a} of the rule ρ_{a} to the l-th premise $S_{\mathrm{a}*l}$ of that rule is expressed by a primitive recursive formula $Correct(\rho_{\mathrm{a}}, \mathrm{a}, l)$. For example

$$Correct(\mathbf{F}, \mathrm{a}, l) :\equiv (l > 0 \to (\exists R \in S_{\mathrm{a}})(S_{\mathrm{a}} = R, S_{\mathrm{a}*l} \& R \in \mathsf{FALSE})$$

B

$$Correct(\forall, \mathrm{a}, l) :\equiv l > 0 \to\ (\exists F, x, \Gamma : S_{\mathrm{a}} = \forall x F[x], \Gamma) S_{\mathrm{a}*l} = F[l-1], \Gamma$$

The following statement is not used below, but forms essential background.

Lemma 1. *$\mathcal{T}_S$ is well-founded iff S is true in the standard model.*

Proof. Standard. □

From now on $\mathcal{T}_\Gamma$ means the proof-search tree of Γ in BI_0^∞.

We use below the following result from [7].

Theorem 1. *For every cut-free derivation $d : \Delta$ of an artimetical sequent Δ in first order arithmetic there is a (primitive-recursive) order preserving embedding 1of d into $\mathcal{T}_\Delta$ establishing that*

$$|\mathcal{T}_\Delta| \leq |d| \cdot \omega$$

for the order-types of (infinite) trees d and $\mathcal{T}_\Delta$.

As a consequence $\mathcal{T}_\Delta$ is a derivation of an arithmetical sequent Δ if anything is, that is iff Δ is derivable.

3 System $\mathrm{BI}_1^{\mathrm{c}}$

The variable q in Ω-rule ranges over arbitrary derivations in BI_0^∞.

Definition 2. *Rules $\Omega^c, \tilde{\Omega}^c$ are obtained from rules $\Omega, \tilde{\Omega}$ by using all arithmetical sequents to index premises.*

$$\frac{\ldots, \Gamma, \Delta, \neg\forall X A(X) \ldots \quad \Delta \in \mathcal{A}}{\Gamma, \neg\forall X A(X)} \; \Omega^c_{\neg\forall X A(X)} B$$

$$\frac{\Gamma, A(Y) \quad \ldots \Gamma, \Delta \ldots \quad \Delta \in \mathcal{A}}{\Gamma} \; \tilde{\Omega}^c_{\neg\forall X A(X)}$$

In more detail, rule Ω^c has a premise $\Gamma, \Delta, \neg\forall X A(X)$ for every (not necessary derivable) arithmetical sequent Δ, while $\tilde{\Omega}^c$ has a premise Γ, Δ for every such Δ plus a premise $\Gamma, A(Y)$. Standard proviso for eigenvariables is assumed.

System $\mathrm{BI}_1^{\mathrm{c}}$ is obtained from the system BI_1^∞ from [4] by replacing rules $\Omega, \tilde{\Omega}$ by $\Omega^c, \tilde{\Omega}^c$. The remaining rule of $\mathrm{BI}_1^{\mathrm{c}}$ are familiar Tait-style rules for $\wedge, \vee, \exists x$, and ω-rule for $\forall x$.

Recall the following definitionl from [8] or [6].

Definition 3. *A* local correct *figure according to given axioms and rules is a tree T of finite sequences* a *of natural numbers ordered by inclusion (so that the empty sequence is the root) and labeled by sequents S_{a} in such a way that for every* $\mathrm{a} \in T$ *one of the following conditions is satisfied:*

1. S_{a} is an axiom and $\mathrm{a} * n \notin T$ for all n,
2. the figure
$$\frac{S_{\mathrm{a}*0} \quad S_{\mathrm{a}*1} \ldots}{S_{\mathrm{a}}}$$
is a correct inference according to one of the rules.

Definition 4. *A* quasi-derivation *of a sequent Γ in* $\mathrm{BI}_1^{\mathrm{c}}$ *is a locally correct tree according to rules of* $\mathrm{BI}_1^{\mathrm{c}}$ *with the root Γ.*

Definition 5. *For any quasi-derivation d in* $\mathrm{BI}_1^{\mathrm{c}}$ *we denote by d^- the result of deleting all premises corresponding to sequents Δ such that $\Delta, A(Y)$* is not derivable in *$BI_0^\infty$from all $\Omega^c_{\neg\forall X A(X)}$-inferences and from all $\tilde{\Omega}^c_{\neg\forall X A(X)}$-inferences. Together with such premise all sequents above it are 1also deleted.*

A quasi-derivation d is a derivation *if d^- is well-founded.*

Note that the operation d^- is continuous, but not effective.

4 Cut-Elimination Operations for $\mathrm{BI}_1^{\mathrm{c}}$

Cut-elimination operations $\mathcal{R}_C, \mathcal{E}, \mathcal{S}, \mathcal{D}_0$ on derivations in BI_1^∞ are defined in [4] by transfinite induction on derivations.

$\mathcal{R}_A$ does one-step reduction of cuts Cut_A over A: the cut is moved up the derivation till it meets an axiom (and disappears) or it meets logical rules introducing A in both premises (then the Cut is replaced by "smaller" cuts), or $A \equiv \forall ZB$ introduced by $\forall X$-rule and $\mathcal{R}_A$ meets Ω introducing $\neg\forall X A(X)$. Then $\tilde{\Omega}$ is introduced.

$\mathrm{B}\mathcal{E}(d)$: applying $\mathcal{R}$ one time to every Cut in the derivation d.

$\mathcal{D}_0$: collapsing of a cut-free derivation possibly containing $\tilde{\Omega}$. Operation $\mathcal{D}_0$ replaces every $\tilde{\Omega}$ by the result of collapsing to a suitable premise (see below).

$\mathcal{S}$ or more precisely $\mathcal{S}_X^{\mathcal{F}}$ for a second order variable X and a formula $\mathcal{F}(x)$ is the result of replacing all occurrences of atomic formulas Xt (where X is free) by $\mathcal{F}(t)$.

However these operations can be defined and proved to satisfy the same recursive equations by primitive recursion (one can say co-recursion) over arbitrary local correct proof figures. For $\mathcal{R}$ and $\mathcal{E}$ this was original definition in [8] developed and extended in [4], [3]. Strictly speaking, one has to explain what "primitive recursive" means for uncountably branching proof figures like BI_1^∞-derivations. Finitary notation system developed in detail in [3] is one of possible explanations. In the present paper we go around the problem of uncountability by using potentially non-well-founded 1but countably branching quasi-derivations.

Below we use the symbols $\mathcal{R}_C, \mathcal{E}, \mathcal{S}$ to denote primitive-recursive operations on local correct figures satisfying the same equations as in Theorems 1 and 2 from [4]. The only essential modification is needed in the definition of the operation $\mathcal{D}_0$. Its intended domain consists of cut-free quasi-derivations of arithmetical sequents. Such quasi-derivation consists of arithmetical sequents, and hence does not contain Ω-inferences. The $\tilde{\Omega}$-rule is collapsed as follows:

$$\frac{\Gamma, A(Y) \quad \ldots d_\Delta : \Gamma, \Delta \ldots}{\Gamma}\ \tilde{\Omega}^c_{\neg\forall X A(X)} \quad \begin{matrix}\text{goes}\\ \text{to}\end{matrix} \quad \frac{\Gamma}{\Gamma}\ \mathrm{Rep}, \tag{3}$$

so that formally there is even no dependence on the derivation of $\Gamma, A(Y)$. In fact we retain exactly the premise corresponding to $\Delta = \Gamma$.

Using notation of [4], Theorem 2 we have B

$$d = \tilde{\Omega}^Y_{\neg P}(d_i)_{i \in \{0\} \cup |P|}$$

and

$$\mathcal{D}_0(d) = \mathrm{Rep}(\mathcal{D}_0(d_\Delta)) \text{ with } \Delta = \Gamma, A(Y).$$

Lemma 2. *1. If $d : \Delta$ is a cut-free quasi-derivation, then $\mathcal{D}_0(d)$ is a quasi-derivation of the same sequent containing none of Cut, $\Omega^c, \tilde{\Omega}^c$.*
2. If in addition d is a derivation then $\mathcal{D}_0(d)$ is a derivation in BI_0^∞.

Proof. 1. This part is easy. Only $\tilde{\Omega}^c$-inferences (3) are changed and countably branching (local correct) tree is replaced by one of its branches, and hence the result is local correct. 2. This is proved by transfinite induction on the well-founded part d^- of d. The main case when d ends in a $\tilde{\Omega}^c$-inference is treated using Lemma 1. By induction hypothesis, $\mathcal{D}_0(d_0)$ is a derivation of $\Gamma, A(Y)$.

Again by induction hypothesis, the quasiderivation d_Γ is a derivation with the ensequent Γ, Γ, that is Γ since sequents are sets of formulas. Application of Rep finishes the proof. □

Lemma 3. *Operations $\mathcal{R}_C, \mathcal{E}, \mathcal{S}$ and $\mathcal{D}_0$ transform (quasi)derivations in* $\mathrm{BI}_1^{\mathrm{c}}$ *into (quasi)derivations.*

Proof. For $\mathcal{D}_0$ this is just proved. For remaining operations the same induction as in [4] (but applied to d^-) goes through.

5 Embedding Finite Derivations into $\mathbf{BI}_1^{\mathbf{c}}$

Finitary system BI_1^- in [4] contains ordinary cut-free Brules of second order arithmetic (for the language restricted as above) plus mathematical induction and the Cut rule (denoted there by R_C). The rules of BI_1^- in Buchholz's notation are as follows:

$$\frac{}{\Delta}\ \mathrm{Ax}^*_\Delta$$

if $\Delta = \{A\} \subseteq \mathrm{TRUE}_0$ or $\Delta = \{C, \neg C\}$

$$\frac{A_0 \quad A_1}{A_0 \& A_1}\ \bigwedge\nolimits_{A_0 \& A_1} \qquad \frac{A_k}{A_0 \vee A_1}\ \bigvee\nolimits_{A_0 \vee A_1}\ (k \in \{0,1\})$$

$$\frac{A(y)}{\forall x A(x)}\ \bigwedge\nolimits^{y}_{\forall x A(x)} \qquad \frac{A(k)}{\exists x A(x)}\ \bigvee\nolimits^{k}_{\exists x A}$$

$$\frac{A(Y)}{\forall X A(X)}\ \bigwedge\nolimits^{Y}_{\forall X A(X)}1 \qquad \frac{A(F)}{\exists X A(X)}\ \bigvee\nolimits^{F}_{\exists X A(X)}$$

$$\frac{\neg F, F(y/Sy)}{\neg F(y/0), F(y/t)}\ \mathrm{Ind}^{y,t}_F \qquad \frac{C \quad \neg C}{\emptyset}\ \mathrm{Cut}_C$$

We introduce operation $^\infty$ transforming every derivation h of a closed endsequent in BI_1^- into a derivation h^∞ of the same sequent in $\mathrm{BI}_1^{\mathrm{c}}$. The only difference from the definition of h^∞ in [4] is the treatment of the rule $\vee_P^{\mathcal{F}}$, that is the rule for $\neg\forall X A(X)$. Instead of expanding it over all possible arithmetical derivations we take for each arithmetical Δ just one quasi-derivation, namely $\mathcal{T}_{\Delta, A(X)}$ when $P = \neg\forall X A(X)$.

A derivation h is called good in section 3 of [4] if it satisfies standard purity conditions on eigenvariables needed for cut-elimination.

So the $\vee_P^{\mathcal{F}}$ clause of our definition for h^∞ looks as follows:

$$(\vee_P^{\mathcal{F}} h_0)^\infty := \Omega_{\neg P}(\mathcal{R}_{P[\mathcal{F}]}(\mathcal{S}_X^{\mathcal{F}}(\mathcal{T}_{\Delta, \neg P[X]}), h_0^\infty))_\Delta$$

where Δ ranges over all arithmetical sequents (cf. the figure in the proof of the next lemma).

Lemma 4. *If h is a good derivation of a closed sequent in* BI_1^- *then h^∞ is a derivation of the same sequent in* $\mathrm{BI}_1^{\mathrm{c}}$*.*

Proof. Local correctness of h^∞ and well-foundedness of $h^{\infty-}$ are easily proved by induction on (finite) derivation h. The main case: h ends in $\neg\forall X$-inference.

$$\frac{h_0 : \Gamma, \neg\forall X A(X), \neg A(\mathcal{F})}{h : \Gamma, \neg\forall X A(X)} \qquad \begin{array}{c} \vdots\ X \\ \mathcal{T}_{\Delta, A(X)} \end{array}$$

$$\begin{array}{c} \vdots\ \mathcal{F} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \\ \mathcal{S}^{\mathcal{F}}_X(\mathcal{T}_{\Delta,A(X)}) :\ \Delta, A(\mathcal{F}) \quad h_0^\infty :\ \Gamma, \neg\forall X A(X), \neg A(\mathcal{F}) \\ \hline\hline h^\infty : \Gamma, \neg\forall X A(X) \end{array}\ \mathcal{R}_{A(\mathcal{F})}$$

$h_0^{\infty-}$ is well-founded by induction hypothesis. For other premises with derivable $\Delta, A(X)$ the figure $\mathcal{T}_{\Delta,A(X)}$ is well-founded by Theorem 1, hence $\mathcal{S}^{\mathcal{F}}_X(\mathcal{T}_{\Delta,A(X)})$ is well-founded by Lemma 3. The same holds for the final application of the operation $\mathcal{R}_{A(\mathcal{F})}$. □

Note. It may seem that the argument above uses something like correctness of one of our formal systems for arithmetical sequents: if a sequent is derivable, then it is true. This is not the case. Let's assume for definiteness that "well-founded" means "every branch is finite". As explained in [7],[8], from every infinite branch in $\mathcal{T}_\Gamma$ one constructs an infinite branch in any other local correct figure for Γ by Δ^0_2 definition. Hence if all branches in an arithmetical derivation $d : \Gamma$ are finite then all branches in $\mathcal{T}_\Gamma$ are finite.

Theorem 2. *For any derivation h of an arithmetical closed sequent in* BI_1^- *with cut-degree m the figure $\mathcal{D}_0\mathcal{E}^m h^\infty$ is an arithmetical cut-free derivation of the same sequent.*

Proof. Operation $\mathcal{E}$ decreases cut-degree at least by 1, operation $\mathcal{D}_0$ eliminates remaining $\tilde{\Omega}^c$-inferences. □

References

1. Beklemishev, L.: Reflection principles and provability algebras in formal arithmetic. Usp. Matem. Nauk 60(2), 3–78 (2005); (in Russian. English translation: Russian Math. Surv. 60(2), 197–268 (2005))
2. Buchholz, W.: The $\Omega_{\mu+1}$-rule. In: Buchholz, W., Feferman, S., Pohlers, W., Sieg, W. (eds.) LNM, vol. 897, pp. 188–233. Springer, Heidelberg (1981)
3. Buchholz, W.: Notation Systems for infinitary Derivations. Arch. Math. Log. 30, 277–296 (1991)
4. Buchholz, W.: Explaining the Gentzen-Takeuti Reduction Steps. Arch. Math. Log. 40, 255–272 (2001)
5. Buchholz, W.: Relating ordinals to proofs in a perspicious way. In: Lecture Notes in Logic, vol. 15, pp. 37–59 (2002)
6. Kreisel, G., Mints, G., Simpson, S.: The Use of Abstract Language in Elementary Metamathematics. In: LNM, vol. 253, pp. 38–131. Springer, Heidelberg (1975)

7. Mints, G.: The Universality of the Canonical Tree. Soviet Math. Dokl. 14, 527–532 (1976)
8. Mints, G.: Finite Investigations of Transfinite Derivations. In: [9], pp. 17–72; (Russian original, 1974)
9. Mints, G.: Selected Papers in Proof Theory. Studies in Proof Theory, Monographs 3. Bibliopolis, Napoli (1992)

Decomposing the Lattice of Meaningless Sets in the Infinitary Lambda Calculus

Paula Severi and Fer-Jan de Vries

Department of Computer Science, University of Leicester, UK

Abstract. The notion of a meaningless set has been defined for infinitary lambda calculus axiomatically. Standard examples of such sets are sets of terms that have no head normal form, the set of terms without weak head normal form and the set of rootactive terms. In this paper, we study the way the intervals decompose as union of more elementary ones. We also analyse the distribution of the sets of meaningless terms in the lattice by selecting some sets as key vertices and study the cardinality in the intervals between key vertices. As an application, we prove that the lattice of meaningless sets is neither distributive nor modular. Interestingly, the example translates into a simple counterexample that the lattice of lambda theories is not modular.

1 Introduction

Classical, finite lambda calculus [1] considers only finite terms. It can not express inside the calculus that certain terms have an infinite normal form. For example, the term MM where $M = \lambda x.f(xx)$ has the infinite normal form $f(f(f(\ldots)))$ which is the limit of the reduction sequence $MM \rightarrow_\beta f(MM) \rightarrow_\beta f(f(MM)) \rightarrow_\beta \ldots$. Infinitary lambda calculus aims to treat finite and infinite terms in one notational framework with notation for finite and infinite reductions. It allows us to express that the above reduction sequence has the infinite term f^ω as limit. However, the natural extension of finite lambda calculus with infinite terms and infinite reductions ruins the confluence property [7]. For example, the term NN, where $N = \lambda x.\mathbf{I}(xx)$ and $\mathbf{I} = \lambda x.x$ reduces both to $\mathbf{I}^\omega$ and $\mathbf{\Omega} = (\lambda x.xx)(\lambda x.xx)$, which can only reduce to themselves and not be joined by even infinite reductions.

Needed to restore the confluence property [7,6,8,5] is a designated set of meaningless terms (for short meaningless set), that is, a set satisfying the Axioms of Meaninglessness [10,5] together with a new rewrite rule that allows any meaningless terms to be rewritten to a fresh symbol $\bot$. Those Axioms are general assumptions needed to prove confluence of the infinitary lambda calculus [10,5]. By changing the meaningless set, we obtain different notions of $\bot$-reduction and different infinite extensions. Each of these extensions is normalising and confluent, so that the set of its normal forms becomes a model of lambda calculus.

A standard example of a meaningless set is the set $\overline{\mathcal{HN}}$ of terms without head normal form. The normal forms of the corresponding infinitary extension

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 210–227, 2011.

$$\begin{tikzcd}
 & & & \Lambda^\infty \arrow[d, "2"] & \\
 & & & \mathcal{HA}\cup\mathcal{IL}\cup\mathcal{O}=\overline{\mathcal{HN}} \arrow[dl, "2"'] \arrow[dr, "2^{\mathfrak{c}}"] & \\
 & & \mathcal{HA}\cup\mathcal{IL}=\overline{\mathcal{HN}}-\mathcal{O} \arrow[dl, "2"'] \arrow[dr, "2^{\mathfrak{c}}"] & & \mathcal{HA}\cup\mathcal{O} \arrow[dl, "2"'] \\
 & \mathcal{SA}\cup\mathcal{SIL}=\overline{\mathcal{WN}} \arrow[dl, "1"'] \arrow[dd, "2^{\mathfrak{c}}"] \arrow[dr, "2^{\mathfrak{c}}"] & & \mathcal{HA} \arrow[dl, "2"] & \\
\cancel{\mathcal{R}\cup\mathcal{SIL}} \arrow[dr, "2^{\mathfrak{c}}"'] & & \mathcal{SA} \arrow[dl, "2^{\mathfrak{c}}"] & & \\
 & \mathcal{R}=\overline{\mathcal{TN}} & & &
\end{tikzcd}$$

Fig. 1. Lattice of Meaningless Sets extended with an auxiliary vertex. The arrow $U_1 \xrightarrow{n} U_2$ indicates that $U_1 \supset U_2$. The label n shows the cardinality of the class of sets of meaningless terms between U_1 and U_2. NB: of all vertices, the vertex $\mathcal{R}\cup\mathcal{SIL}$ is not a meaningless set.

of finite lambda calculus are precisely the Böhm trees, but now their definition is within the syntax of infinite lambda calculus, whereas [1] needed to develop a special notational machinery. Similarly the choice of the set $\overline{\mathcal{WN}}$ of terms without weak head normal forms as set of meaningless terms leads to the Lévy-Longo trees [6,8,5], and the choice for the set $\mathcal{R}$ of rootactive terms recaptures the Berarducci trees [3,6,8,5]. Although in the initial papers [6,8,10,5] on infinite lambda calculus only those three sample sets were presented as set of meaningless terms, these sets are not the only sets of meaningless terms. Only in the more recent papers [14,15,9] some aspects of the rich lattice of the sets of meaningless terms have been explored.

The set of all sets of meaningless terms forms a complete lattice as depicted in Figure 1. We say $U_1 \to U_2$ when $U_1 \supset U_2$. The bottom element is the set $\mathcal{R}$ and the top element is the set Λ^∞ of finite and $\bot$-free infinite lambda terms. The meet operation $\sqcap$ is intersection and the join $\sqcup$ of two sets of meaningless terms is the smallest meaningless set that contains the two sets.

The purpose of the current paper is to analyse the distribution of the sets of meaningless terms in the lattice by selecting some sets as *key vertices* and study the cardinality in the intervals between key vertices. The key vertices are all depicted in Figure 1. All key vertices stand for sets of meaningless terms except for the vertex $\mathcal{R}\cup\mathcal{SIL}$. We included this set in the figure to provide a complete picture of the lattice. Because, despite the fact that $\mathcal{R}\cup\mathcal{SIL}$ itself is not meaningless, there are infinitely many sets of meaningless terms between $\mathcal{R}$ and $\mathcal{R}\cup\mathcal{SIL}$. The other vertices in Figure 1 represent all sets of meaningless terms that decompose as the disjoint union of one or more of the basic sets $\mathcal{R}$, $\mathcal{SIL}$, $\mathcal{IL}$, $\mathcal{SA}$, $\mathcal{HA}$ and $\mathcal{O}$ [15].

We consider intervals $[U_1, U_2] = \{U \mid U_1 \subseteq U \subseteq U_2 \text{ and } U \text{ is meaningless}\}$ between two arbitrary sets U_1 and U_2. In particular, we will distinguish between *key intervals* which are intervals between any two key vertices and *elementary key intervals* which are key intervals between two consecutive key vertices. We study *the cardinality of the elementary key intervals.* The cardinalities are shown as labels above the arrows of Figure 1. Some intervals have cardinality 2 and contain only both extremes. Only one of them has cardinality 1 which is $[\mathcal{R}\cup\mathcal{SIL}, \mathcal{SA}\cup\mathcal{SIL}]$ which contains only the right extreme. All others are uncountable and have cardinality $2^{\mathfrak{c}}$ where $\mathfrak{c}$ is the cardinality of the continuum.

We show that the elementary key intervals with cardinality $2^{\mathfrak{c}}$ cannot be finitely decomposed. In other words, the uncountable intervals cannot be further decomposed as union of finite subintervals. We also study how the key intervals are *decomposed as union of elementary key intervals.* We prove that all key intervals which are above the set $\mathcal{SA}$ and $\mathcal{R}\cup\mathcal{SIL}$ can be decomposed as union of elementary key intervals. For example, $[\mathcal{SA}, \mathcal{HA}\cup\mathcal{O}] = [\mathcal{SA}, \mathcal{HA}]\cup[\mathcal{HA}, \mathcal{HA}\cup\mathcal{O}]$. Not all the key intervals have this nice property. We show that the interval $[\mathcal{R}, \mathcal{SA}\cup\mathcal{SIL}]$ can not be decomposed as union of elementary key intervals. For this, we show that there are $2^{\mathfrak{c}}$ many sets of meaningless terms which are in $[\mathcal{R}, \mathcal{SA}\cup\mathcal{SIL}]$ and are not in either $[\mathcal{R}, \mathcal{SA}]$, $[\mathcal{SA}, \mathcal{SA}\cup\mathcal{SIL}]$ or $[\mathcal{R}, \mathcal{R}\cup\mathcal{SIL}]$. This is depicted in Figure 1 by an arrow from $\mathcal{SA}\cup\mathcal{SIL}$ to $\mathcal{R}$ labelled by $2^{\mathfrak{c}}$.

We conclude with the observation that the lattice of sets of meaningless sets is neither modular nor distributive. Interestingly, the example translates into a simple counterexample that the lattice of lambda theories is not modular.

2 Infinitary Lambda Calculus

We will now briefly recall some notions and facts of infinitary lambda calculus from our earlier work [6,8,5,13,16]. We assume familiarity with basic notions and notations from [1]. Let Λ be the set of λ-terms and $\Lambda_\perp$ be the set of finite λ-terms with $\perp$. The set $\Lambda^\infty_\perp$ of finite and infinite λ-terms is defined by coinduction from the grammar: $M ::= \perp \mid x \mid (\lambda x M) \mid (MM)$ where x is a variable from some fixed set of variables $\mathcal{V}$. The set Λ^∞ is the subset of $\perp$-free terms. The set $(\Lambda^\infty)^0$ is the subset of closed terms (without free variables) in Λ^∞.

We follow the usual conventions on syntax. We will also use the following abbreviations for terms:

$$\begin{array}{lll} \mathbf{I} = \lambda x.x & \mathbf{O} = \lambda x_1.\lambda x_2.\lambda x_3.\ldots & {}^{\omega}M = (((\ldots)M)M)M \\ \mathbf{K} = \lambda xy.x & \mathbf{\Omega} = (\lambda x.xx)(\lambda x.xx) & \mathbf{Fix} = (\lambda xy.y(xxy))(\lambda xy.y(xxy)) \end{array}$$

The set $\Lambda^\infty_\perp$ contains the three sets of Böhm, Lévy–Longo and Berarducci trees. These notions are usually pictured visually as trees but up to some change of representation they are terms belonging to $\Lambda^\infty_\perp$. So, in the following we will freely identify trees with terms in $\Lambda^\infty_\perp$. In [8,10,5], an alternative definition of the set $\Lambda^\infty_\perp$ is given using a metric. The coinductive and metric definitions are equivalent [2]. We define the β-rule on the set $\Lambda^\infty_\perp$ of finite and infinite terms:

$$(\lambda x.M)N \to M[x := N] \quad (\beta)$$

The reduction $\rightarrow_\beta$ is defined as the smallest binary relations containing β which is closed under contexts. The β_h-reduction is the restriction of the β-reduction to head redexes. Let $U \subseteq \Lambda^\infty$ where Λ^∞ is the set of terms in $\Lambda^\infty_\perp$ that do not contain $\perp$. We define the $\perp_U$-rule rule on $\Lambda^\infty_\perp$ as follows:

$$\frac{M[\perp := \mathbf{\Omega}] \in U \quad M \neq \perp}{M \rightarrow \perp}\,(\perp_U)$$

When there is no danger of confusion, we denote $\perp_U$ by $\perp$. The reduction $\rightarrow_{\beta\perp_U}$ is defined as the smallest binary relation containing β and $\perp_U$ which is closed under contexts.

Each set U of meaningless terms gives rise to a different infinitary lambda calculus. The infinitary lambda calculus λ^∞_U on U is the calculus $\lambda^\infty_U = (\Lambda^\infty_\perp, \rightarrow_{\beta\perp_U})$.

In infinitary lambda calculus we consider strongly converging reduction sequences. These can be of any countable, transfinite length α: $M_0 \rightarrow_\rho M_1 \rightarrow \ldots M_\omega \rightarrow M_{\omega+1} \rightarrow \ldots M_{\omega+\omega} \rightarrow M_{\omega+\omega+1} \rightarrow \ldots M_\alpha$, where $\rightarrow$ stands for a β- or $\perp$-reduction step. The rough idea is that in such reductions for each limit ordinal λ, the term M_λ is defined as the Cauchy limit of the preceding reduction. These limits can then be further reduced. In addition the depth of the contracted redexes goes to infinity at each limit term. We use the following notation: $M \rightarrow N$ denotes a one step reduction from M to N; $M \twoheadrightarrow N$ denotes a finite reduction from M to N; $M \twoheadrightarrow\!\!\!\!\rightarrow N$ denotes a strongly converging reduction from M to N. When λ^∞_U is confluent and normalizing, the normal form of a term M in λ^∞_U is denoted by $\mathsf{nf}_U(M)$. Each confluent and normalising λ^∞_U gives rise to a λ-model and a λ-theory of the finite lambda calculus.

Definition 1. *Let $U \subseteq \Lambda^\infty$ and λ^∞_U be confluent and normalising.*

1. *The λ-model $\mathfrak{M}_U$ induced by the infinintary lambda calculus λ^∞_U is defined as follows. The domain of $\mathfrak{M}_U$ is the set $\mathsf{nf}_U(\mathit{\Lambda})$ of normal forms of finite terms. We interpret a lambda term $M \in \mathit{\Lambda}$ by its normal form $\mathsf{nf}_U(M)$ and we define application simply by $\mathsf{nf}_U(M) \bullet \mathsf{nf}_U(N) = \mathsf{nf}_U(M \bullet N)$.*
2. *The λ-theory induced by the infinitary lambda calculus λ^∞_U is denoted by $\mathfrak{T}_U$ and defined as $\mathfrak{T}_U = \{M = N \mid M, N \in \mathit{\Lambda}^0 \ \&\ \mathsf{nf}_U(M) = \mathsf{nf}_U(N)\}$.*

It is easy to show that $\mathfrak{M}_U$ is a λ-model and $\mathfrak{T}_U$ is a λ-theory of the finite lambda calculus [1,11]. The set of λ-theories of the finite lambda calculus form a complete lattice where the meet $\sqcap$ is intersection and the join $\sqcup$ of a family of sets is the smallest theory that containts the family.

We will now define the notion of set of meaningless term. We follow the definition of [9]. This definition differs slightly from the earlier definition in [8,10,5] in that the axiom of closure under β-expansion has been added. This addition has a number of useful consequences. The first is, as observed in [9], that with the extra axiom the calculus λ^∞_U is not only confluent and normalising, but also ω-compressible. The second is that for any set $U \subseteq \Lambda^\infty$, the infinitary lambda calculi λ^∞_U and $\lambda^\infty_{\widehat{U}}$ induce the same reduction relations $\twoheadrightarrow\!\!\!\!\rightarrow_{\beta\perp_U}$ and

$\twoheadrightarrow_{\beta\bot_{\widehat{U}}}$ where $\widehat{U} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N \ \&\ N \in U\}$. Hence, if λ_U^∞ is confluent and normalizing, we have that $\mathfrak{M}_U = \mathfrak{M}_{\widehat{U}}$ and $\mathfrak{T}_U = \mathfrak{T}_{\widehat{U}}$. This last consequence is pertinent for the current paper.

Notation 2. *Let $M \overset{U}{\leftrightarrow} N$ denote that N is obtained from M by replacing some (possibly infinitely many) subterms in U by other terms in U.*

Definition 3. *[10,5]* We say that $U \subseteq \Lambda^\infty$ is a *set of meaningless terms* (also called a *meaningless set*) if it is a set satisfying the *axioms of meaninglessness*:

1. Rootactiveness. $\mathcal{R} = \{M \in \Lambda^\infty \mid M$ is rootactive$\} \subseteq U$ (see Definition 5).
2. Closure under β-reduction. For all $M \in U$, if $M \twoheadrightarrow_\beta N$ then $N \in U$.
3. Closure under substitution. For all $M \in U$, $M^\sigma \in U$.
4. Overlap. For all $\lambda x.M \in U$, $(\lambda x.M)N \in U$.
5. Indiscernibility. For all $M, N \in \Lambda^\infty$, $M \in U$ if and only if $N \in U$.
6. Closure under β-expansion. For all $N \in U$, if $M \twoheadrightarrow_\beta N$, then $M \in U$.

Note that $\mathbf{\Omega} \in U$ for all set U of meaningless terms because $\mathbf{\Omega}$ is rootactive.

Note also that even without requiring closure under β-expansion, we have that meaningless sets contain certain β-expansions: for instance just from rootactiveness and indiscernibility it follows that $\mathbf{I}(\mathbf{I}M) \in U$ and $\mathbf{K}MN \in U$ if $M \in U$.

Theorem 4. *[8,10,5] If U is a set of meaningless terms then λ_U^∞ is confluent and normalising.*

We will now define the sets of meaningless terms that occur in Figure 1. To define these sets, we will first need to introduce new forms of terms analogous to the notions of head, weak head and top normal forms and define certain specific subsets of Λ^∞ containing the respective forms [15].

Definition 5. *We define that*

1. $\mathcal{R} = \{M \in \Lambda^\infty \mid M$ *is rootactive*$\}$ *where M is a* rootactive form *if for all $M \twoheadrightarrow_\beta N$ there exists a redex $(\lambda x.P)Q$ such that $N \twoheadrightarrow_\beta (\lambda x.P)Q$.*
2. $\mathcal{SA} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is a strong active form*$\}$ *where M is a* strong active form *if $M = RP_1 \ldots P_k$ and R is rootactive.*
3. $\mathcal{HA} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is a head active form*$\}$ *where M is a* head active form *if $M = \lambda x_1 \ldots x_n.RP_1 \ldots P_k$ and R is rootactive.*
4. $\mathcal{SIL} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is a strong infinite left spine form* $\}$ *where M is a* strong infinite left spine form *if $M = (\ldots P_2)P_1$.*
5. $\mathcal{IL} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is an infinite left spine form*$\}$ *where M is an* infinite left spine form *if $M = \lambda x_1 \ldots x_n.(\ldots P_2)P_1$.*
6. $\mathcal{HN} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is a head normal form*$\}$ *where M is a* head normal form *(hnf) if $M = \lambda x_1 \ldots x_n.yP_1 \ldots P_k$.*
7. $\mathcal{WN} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and N is a weak head normal form*$\}$ *where M is a* weak head normal form *(whnf) if M is a hnf or $M = \lambda x.N$.*

8. $\mathcal{TN} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ *and* N *is a top normal form*$\}$ *where* M *is a* top normal form *(tnf) if it is either a whnf or an application* (NP) *if there is no* Q *such that* $N \twoheadrightarrow_\beta \lambda x.Q$.
9. $\mathcal{O} = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta \mathbf{O}\}$.

Theorem 6. *[10,15] The sets* $\mathcal{R}$, $\mathcal{SA}$, $\mathcal{HA}$, $\mathcal{HA} \cup \mathcal{O}$, $\mathcal{SA} \cup \mathcal{SIL}$, $\mathcal{HA} \cup \mathcal{IL}$, *and* $\mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}$ *are sets of meaningless terms.*

As already said in the introduction, by $\overline{\mathcal{HN}}$, $\overline{\mathcal{WN}}$ and $\overline{\mathcal{TN}}$ we denote the complements in Λ^∞ of $\mathcal{HN}$, $\mathcal{WN}$ and $\mathcal{TN}$ respectively. As it happens, a term is rootactive if and only if it has no top normal form. Hence $\overline{\mathcal{TN}} = \mathcal{R}$.

Definition 7. *We define the Berarducci tree of a term* M *(denoted by* $\mathsf{BerT}(M)$*) by co-recursion as follows.*

1. $\mathsf{BerT}(M) = \bot$, *if* M *is rootactive.*
2. $\mathsf{BerT}(M) = \lambda x.\mathsf{BerT}(N)$, *if* $M \twoheadrightarrow_\beta \lambda x.N$.
3. $\mathsf{BerT}(M) = \mathsf{BerT}(N)\mathsf{BerT}(P)$ *if* $M \twoheadrightarrow_\beta NP$ *and there is no* Q *such that* $N \twoheadrightarrow_\beta Q$ *and* Q *is an abstraction.*

The Berarducci tree $\mathsf{BerT}(M)$ of a term M is the normal form of M in λ^∞_U where for U we take $\mathcal{R}$ [3,8].

Definition 8. *We define the Lévy-Longo tree of a term* M *(denoted by* $\mathsf{LLT}(M)$*) by co-recursion as follows.*

1. $\mathsf{LLT}(M) = \bot$, *if* M *has no weak head normal form.*
2. $\mathsf{LLT}(M) = y\ \mathsf{LLT}(M_1)\ldots\mathsf{LLT}(M_m)$, *if* $M \twoheadrightarrow_\beta yM_1\ldots M_m$.
3. $\mathsf{LLT}(M) = \lambda x.\mathsf{LLT}(N)$, *if* $M \twoheadrightarrow_\beta \lambda x.N$.

The Lévy Longo tree $\mathsf{LLT}(M)$ of a term M is the normal form of M in the calculus $\lambda^\infty_{\overline{\mathcal{WN}}}$. It is easy to see that $\overline{\mathcal{WN}} = \mathcal{SA} \cup \mathcal{SIL}$ [8].

Definition 9. *We define the Böhm tree of a term* M *(denoted by* $\mathsf{BT}(M)$*) by co-recursion as follows.*

1. $\mathsf{BT}(M) = \bot$, *if* M *has no head normal form and*
2. $\mathsf{BT}(M) = \lambda x_1\ldots\lambda x_n.y\ \mathsf{BT}(M_1)\ldots\mathsf{BT}(M_m)$, *if* M *has a finite* β*-reduction to* $\lambda x_1\ldots\lambda x_n.yM_1\ldots M_m$.

The Böhm tree $\mathsf{BT}(M)$ of a term M is the normal form of M in the calculus $\lambda^\infty_{\overline{\mathcal{HN}}}$. It is easy to see that $\overline{\mathcal{HN}} = \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}$ [1,8].

Notation 10. *Let* $X \subseteq \Lambda^\infty_\bot$. *We use the following notation:* $\mathsf{BerT}(X) = \{\mathsf{BerT}(M) \mid M \in X\}$, $\mathsf{LLT}(X) = \{\mathsf{LLT}(M) \mid M \in X\}$ *and* $\mathsf{BT}(X) = \{\mathsf{BT}(M) \mid M \in X\}$.

Remark 11. Not all (combinations of) basic sets give rise to a meaningless set.

1. The sets $\mathcal{SIL}$, $\mathcal{IL}$ and $\mathcal{O}$ do not satisfy rootactiveness.
2. The sets $\mathcal{R} \cup \mathcal{SIL}$ and $\mathcal{R} \cup \mathcal{IL}$ do not satisfy indiscernibility: The term ${}^{\omega}\mathbf{I} = ((\ldots)\mathbf{I})\mathbf{I}$ belongs to both sets $\mathcal{SIL}$ and $\mathcal{IL}$. Since ${}^{\omega}\mathbf{I} = ({}^{\omega}\mathbf{I})\mathbf{I}$, a set satisfying indiscernibility should contain $\mathbf{\Omega I}$ as well. However, $\mathbf{\Omega I}$ does not belong to neither $\mathcal{SIL}$ nor $\mathcal{IL}$.
3. The set $\mathcal{R} \cup \mathcal{O}$ is not meaningless. Because any set containing $\mathbf{O}$ must contain $\lambda x.\mathbf{\Omega}$ by indiscernibility since $\mathbf{O} = \lambda x.\mathbf{O}$.

Definition 12. *1. The* key vertices *are the sets that appear in Figure 1, i.e.* $\mathcal{R}, \mathcal{SA}, \mathcal{HA}, \mathcal{HA} \cup \mathcal{O}, \mathcal{R} \cup \mathcal{SIL}, \mathcal{SA} \cup \mathcal{SIL}, \mathcal{HA} \cup \mathcal{IL}, \mathcal{HA} \cup \mathcal{O} \cup \mathcal{IL}$ *and* Λ^{∞}*. The set of key vertices is denoted by* $\mathcal{K}$*.*

2. The set $\mathcal{I}_{\mathcal{K}}$ *of* key intervals *is the set of intervals whose extremes are only some of the sets in* $\mathcal{K}$*, i.e.* $\mathcal{I}_{\mathcal{K}} = \{[U_1, U_2] \mid U_1, U_2 \in \mathcal{K}\}$*.*

3. An elementary key interval *is an interval* $[U_1, U_2]$ *in* $\mathcal{K}$ *such that* $U_1 \subset U_2$ *and there is no other set* $U \in \mathcal{K}$ *between* U_1 *and* U_2*.*

The sets in $\mathcal{I}_{\mathcal{K}}$ are all meaningless except for $\mathcal{R} \cup \mathcal{SIL}$. Note that $[\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}]$ is an elementary key interval, but $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$ is not.

3 The Elementary Key Intervals of Finite Cardinality

We will now prove that the intervals $[\mathcal{SA}, \mathcal{HA}]$, $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{O}]$, $[\mathcal{SA} \cup \mathcal{SIL}, \mathcal{HA} \cup \mathcal{IL}]$, $[\mathcal{HA} \cup \mathcal{IL}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$ and $[\mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}, \Lambda^{\infty}]$ contain only the extremes and have cardinality 2 and the interval $[\mathcal{R} \cup \mathcal{SIL}, \mathcal{SA} \cup \mathcal{SIL}]$ has cardinality 1.

Theorem 13. *The interval* $[\overline{\mathcal{HN}}, \Lambda^{\infty}]$ *has cardinality* 2*.*

Proof. By Lemma 45 the only sets in $[\overline{\mathcal{HN}}, \Lambda^{\infty}]$ are $\overline{\mathcal{HN}}$ and Λ^{∞}. □

Theorem 14. *The intervals* $[\mathcal{SA}, \mathcal{HA}]$ *and* $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{O}]$ *have cardinality* 2*.*

Proof. We prove that $\mathcal{HA}$ is the only meaningless set between $\mathcal{SA}$ and $\mathcal{HA} \cup \mathcal{O}$. Suppose there exists a set U of meaningless terms such that $\mathcal{SA} \subset U \subset \mathcal{HA} \cup \mathcal{O}$. Then there exists $M \in (\mathcal{HA} \cup \mathcal{O}) - \mathcal{SA}$. Then M should reduce to a term N either of the form $\mathbf{O}$ or $\lambda x_1 \ldots x_n.RP_1 \ldots P_k$ with $n \geq 1$. By Lemma 46(2), in both cases we have $\mathcal{HA} \subseteq U$. Since, $U \subset \mathcal{HA} \cup \mathcal{O}$, we get $U = \mathcal{HA}$. □

Theorem 15. *The intervals* $[\mathcal{SA} \cup \mathcal{SIL}, \mathcal{HA} \cup \mathcal{IL}]$ *and* $[\mathcal{HA} \cup \mathcal{IL}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$ *have cardinality* 2*.*

Proof. To prove that $\mathcal{HA} \cup \mathcal{IL}$ is the only meaningless set between $\mathcal{SA} \cup \mathcal{SIL}$ and $\mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}$, we follow the proof of Theorem 14 using Lemma 46(3) instead. □

Theorem 16. *The interval* $[\mathcal{R} \cup \mathcal{SIL}, \mathcal{SA} \cup \mathcal{IL}]$ *has cardinality* 1*.*

Proof. It follows from Lemma 47 that the only meaningless set in $[\mathcal{R} \cup \mathcal{SIL}, \mathcal{SA} \cup \mathcal{IL}]$ is $\mathcal{SA} \cup \mathcal{SIL}$. □

As a consequence of Lemma 46 and Theorems 14, 15 and 47, we have that:

Theorem 17. *All the key intervals above $\mathcal{SA}$ and also above $\mathcal{R} \cup \mathcal{SIL}$ can be decomposed as unions of elementary key intervals.*

In particular, we have that:

$$\begin{aligned}
[\mathcal{SA}, \Lambda^\infty] &= [\mathcal{SA}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}] \cup [\mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}, \Lambda^\infty] \\
[\mathcal{SA}, \mathcal{HA} \cup \mathcal{IL}] &= [\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}] \cup [\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}] \\
[\mathcal{SA}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}] &= [\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}] \cup [\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}] \cup [\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}] \\
[\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}] &= [\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}] \cup [\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]
\end{aligned}$$

4 The Indecomposable Key Interval $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$

We will show that the key interval $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$ cannot be decomposed as union of elementary key intervals. For this, we will first show that there are $2^{\mathfrak{c}}$ many sets of meaningless terms in $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}] - ([\mathcal{R}, \mathcal{SA}] \cup [\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}] \cup [\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}])$. As a consequence, we have that

$$[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}] \neq [\mathcal{R}, \mathcal{SA}] \cup [\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}] \cup [\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}]$$

We will also show a stronger property which is that the interval $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$ cannot be finitely decomposed, not even by taking intervals with other extremes apart from the sets of Figure 1.

Definition 18. *Let $M \in \Lambda^\infty$ and $X \subseteq \Lambda^\infty$.*

1. *M is a* strong infinite left spine form relative *to X (X-sil) if $M = ((\ldots)P_2)P_1$ and $P_i \in X$ for all i.*
2. *$\mathcal{SIL}_X = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X-sil$\}$.*

Remark 19. 1. $\mathcal{SIL}_X$ is not a set of meaningless terms since it does not satisfy rootactiveness. Neither $\mathcal{R} \cup \mathcal{SIL}_X$ is a meaningless set since it does not satisfy indiscernibility. Let $M \in X$. The term ${}^{\omega}M = ((\ldots)M)M \in \mathcal{SIL}_X$ but $\mathbf{\Omega} M$ does not belong to $\mathcal{R} \cup \mathcal{SIL}_X$.
2. $\mathcal{SA} \cup \mathcal{SIL}_X$ is not a meaningless set since it does not satisfy indiscernibility. Consider a term $P \in \Lambda^\infty - X$ and $M \in \mathcal{SIL}_X$. The term $\mathbf{\Omega} P \in \mathcal{SA}$ but $MP \notin \mathcal{SIL}_X$.

The above remark motivates the following definition:

Definition 20. *Let $M \in \Lambda^\infty$ and $X \subseteq \Lambda^\infty$.*

1. *M is a* strong active form relative *to X (X-saf) if $M = RP_1 \ldots P_k$ and R is rootactive and $P_1, \ldots, P_k \in X$.*
2. *$\mathcal{SA}_X = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X-saf$\}$.*

Theorem 21. [9] *Let $X \subseteq \mathsf{LLT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$. Then, $\mathcal{SA}_X \cup \mathcal{SIL}_X$ is a set of meaningless terms.*

Corollary 22. *There are $2^{\mathfrak{c}}$ many sets of meaningless terms between $\mathcal{R}$ and $\mathcal{SA} \cup \mathcal{SIL}$ which are not in either $[\mathcal{R}, \mathcal{SA}]$, or $[\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}]$ or $[\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}]$.*

Corollary 23. *The interval $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$ is not finitely decomposable.*

Proof. Clearly, the class $\{\mathcal{SA}_X \cup \mathcal{SIL}_X \mid X$ is singleton$\}$ of meaningless sets are all unrelated to each other. They all appear in "parallel intervals". □

5 The Elementary Key Intervals of Infinite Cardinality

We will now show that the intervals $[\mathcal{R}, \mathcal{SA}]$, $[\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}]$, $[\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}]$, $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}]$, and $[\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$ have cardinality $2^{\mathfrak{c}}$ where $\mathfrak{c}$ is the cardinality of the continuum. We can deduce that all these intervals are not finitely decomposable by taking singleton sets as in the proof of Corollary 23.

5.1 The Interval $[\mathcal{R}, \mathcal{SA}]$

We will show that there are $2^{\mathfrak{c}}$ sets of meaningless terms between $\mathcal{R}$ and $\mathcal{SA}$.

Theorem 24. *[15] Let $X \subseteq \mathsf{BerT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$. Then, $\mathcal{SA}_X$ is a set of meaningless terms.*

Corollary 25. *The interval $[\mathcal{R}, \mathcal{SA}]$ has cardinality $2^{\mathfrak{c}}$ and is not finitely decomposable.*

5.2 The Interval $[\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}]$

To build a set U of meaningless terms between $\mathcal{R}$ and $\mathcal{SIL}$, we have to exclude from U those strong infinite left spines that are prefix of themselves. For instance the assumption $((\ldots)\mathbf{I})\mathbf{I} \in U$ that would otherwise imply $\mathbf{\Omega I} \in U$ (see Remark 11). The set $\mathcal{R} \cup \{((\ldots)\mathbf{I})\mathbf{I})\mathbf{K}\}$ is a set of meaningless terms but $\mathcal{R} \cup \{((\ldots)\mathbf{I})\mathbf{I}\}$ is not.

Definition 26. *Let $M \in \Lambda^\infty$ and $X, Y \subseteq \Lambda^\infty$.*

1. *M is a* strong infinite left spine form relative *to X and Y (X, Y-silf) if $M = NP$ where N is a strong infinite left spine relative to X and $P \in Y$.*
2. *$\mathcal{SIL}_X^Y = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X, Y-silf$\}$.*

Theorem 27. *Let $X, Y \subseteq \mathsf{LLT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$ and $X \cap Y = \emptyset$. Then, $\mathcal{R} \cup \mathcal{SIL}_X^Y$ is a meaningless set.*

Corollary 28. *The interval $[\mathcal{R}, \mathcal{R} \cup \mathcal{SIL}]$ has cardinality $2^{\mathfrak{c}}$ and is not finitely decomposable.*

5.3 The Interval $[\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}]$

Let U be a meaningless set. As $\mathbf{\Omega} P_1 \ldots P_n \in \mathcal{SA} \subset U$ we obtain from indiscernibility that $MP_1 \ldots P_n \in U$ for any $M \in U$ and $P_1, \ldots P_n \in \Lambda^\infty$ This motivates:

Definition 29. *Let $M \in \Lambda^\infty$ and $X \subseteq \Lambda^\infty$.*

1. *M is a* segmented strong infinite left spine form relative *to X (X-ssf) if there exists a finite set $\{P_1, \ldots, P_n\} \subseteq \Lambda^\infty_\perp$ (possible empty) such that $M = NP_1 \ldots P_n$ and N is a strong infinite left spine relative to X.*
2. *$\mathcal{SS}_X = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X-ssf$\}$.*

Theorem 30. *[9] Let $X \subseteq \mathsf{LLT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$. Then, $\mathcal{SA} \cup \mathcal{SS}_X$ is a meaningless set.*

Corollary 31. *The interval $[\mathcal{SA}, \mathcal{SA} \cup \mathcal{SIL}]$ has cardinality $2^{\mathfrak{c}}$ and is not finitely decomposable.*

5.4 The Intervals $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}]$ and $[\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$

A set U of meaningless terms containing $\mathcal{HA}$ is closed under arbitrary applications and abstractions, i.e. if $M \in U$ and $P_1, \dots P_n \in \Lambda^\infty$ we should also have that $\lambda x_1 \dots x_k.MP_1 \dots P_n \in U$ because $\lambda x_1 \dots x_k.\boldsymbol{\Omega} P_1 \dots P_n \in \mathcal{HA} \subset U$. This motivates the definition:

Definition 32. *Let $M \in \Lambda^\infty$ and $X \subseteq \Lambda^\infty$.*

1. *M is a* segmented infinite left spine form relative *to X (X-sf) if there exists a finite set $\{P_1, \dots, P_n\} \subseteq \Lambda^\infty_\perp$ (possible empty) such that $M = \lambda x_1 \dots x_k.NP_1 \dots P_n$ and N is a strong infinite left spine relative to X.*
2. *$\mathcal{S}_X = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X-sf$\}$.*

The first item of the following theorem is as Theorem 47 in [9] but the hypothesis of the second item has been restricted.

Theorem 33. *The following sets are sets of meaningless terms:*

1. *$\mathcal{HA} \cup \mathcal{S}_X$ provided $X \subseteq \mathsf{LLT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$.*
2. *$\mathcal{HA} \cup \mathcal{O} \cup \mathcal{S}_X$ provided $X \subseteq \mathsf{BT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$.*

Corollary 34. *The intervals $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}]$ and $[\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$ have cardinality $2^{\mathfrak{c}}$ and they are not finitely decomposable.*

6 Non-modularity and Non-distributivity

In this section we prove that the lattice of meaningless sets is neither modular nor distributive by applying the M_3-N_5 Theorem of [4] and the previous theory.

Definition 35. *Let $M \in \Lambda^\infty$ and $X \subseteq \Lambda^\infty$.*

1. *M is a* segmented ${}^\omega\mathbf{KI}$-term relative *to X (X-stf) if there exists a finite set $\{P_1, \dots, P_n\} \subseteq X$ (possible empty) such that $M = {}^\omega\mathbf{KI}P_1 \dots P_n$.*
2. *$\mathcal{KI}_X = \{M \in \Lambda^\infty \mid M \twoheadrightarrow_\beta N$ and N is a X-stf$\}$.*

Lemma 36. *Let $X \subseteq \mathsf{LLT}(\Lambda^\infty) \cap (\Lambda^\infty)^0$. Then, $\mathcal{SA}_X \cup \mathcal{KI}_X$ is a meaningless set.*

Theorem 37. *The lattice of sets of meaningless sets is neither modular nor distributive.*

Proof. The key interval $[\overline{\mathcal{TN}}, \mathcal{SA} \cup \mathcal{SIL}]$ contains a sublattice isomorphic to N_5.

$$\begin{tikzcd}
 & U_5 = \mathcal{SA}_{\{\mathbf{I},\mathbf{K}\}} \cup \mathcal{KI}_{\{\mathbf{I},\mathbf{K}\}} \arrow[dl] \arrow[ddr] & \\
U_4 = \mathcal{SA}_{\{\mathbf{I}\}} \cup \mathcal{KI}_{\{\mathbf{I}\}} \arrow[dd] & & \\
 & & U_3 = \mathcal{SA}_{\{\mathbf{K}\}} \cup \mathcal{SIL}_{\{\mathbf{K}\}} \arrow[ddl] \\
U_2 = \mathcal{R} \cup \mathcal{SIL}^{\{\mathbf{I}\}}_{\{\mathbf{K}\}} \arrow[dr] & & \\
 & U_1 = \mathcal{R} &
\end{tikzcd}$$

By Theorem 27, $U_2 = \mathcal{R} \cup \mathcal{SIL}^{\{\mathbf{I}\}}_{\{\mathbf{K}\}}$ is the smallest meaningless set closed under β-expansions containing ${}^{\omega}\mathbf{KI}$. By Theorem 21, $U_3 = \mathcal{SA}_{\{\mathbf{K}\}} \cup \mathcal{SIL}_{\{\mathbf{K}\}}$ is the smallest meaningless set closed under β-expansions containing $\mathbf{\Omega K}$ and ${}^{\omega}\mathbf{K}$. By Theorem 36, $U_4 = \mathcal{SA}_{\{\mathbf{I}\}} \cup \mathcal{KI}_{\{\mathbf{I}\}}$ is the smallest meaningless set that is closed under β-expansions and contains $\mathbf{\Omega I}$ and ${}^{\omega}\mathbf{KI}$. By Theorem 36, $U_5 = \mathcal{SA}_{\{\mathbf{I},\mathbf{K}\}} \cup \mathcal{KI}_{\{\mathbf{I},\mathbf{K}\}}$ is the smallest meaningless set closed under β-expansions containing $\mathbf{\Omega I}$, $\mathbf{\Omega K}$ and ${}^{\omega}\mathbf{KI}$.

To prove that the above five sets form a sublattice of the lattice of sets of meaningless terms, we have to prove that the sublattice is closed under the join and meet operations, i.e. $U_5 = U_3 \sqcup U_4$ and $U_1 = U_2 \sqcap U_3$. The latter is trivial because the meet $\sqcap$ is intersection. For the first equation, it is not difficult to show that U_5 is the smallest set of meaningless terms that contains U_3 and U_4. □

Corollary 38. *Let U_1, U_2, U_3, U_4 and U_5 be sets of meaningless terms as used in the proof of Theorem 37.*

1. *For all $1 \leq i \leq 5$, the infinitary lambda calculus $\lambda^{\infty}_{U_i}$ is confluent and normalising.*
2. *For all $1 \leq i \leq 5$, the theory $\mathfrak{T}_{U_i}$ induced by $\lambda^{\infty}_{U_i}$ is consistent.*

Part (1) follows from Theorem 4. Part (2) follows from the fact that these five calculi have at least two different normal forms which are $\mathbf{I}$ and $\mathbf{K}$.

In the following lemma, P^n denotes the truncation of P at depth n.

Lemma 39. *Let U_2 and U_3 be the sets of meaningless terms used in the proof of Theorem 37. For all n, $P, Q \in \mathsf{BerT}(\Lambda^{\infty}_{\perp})$, if $\mathsf{nf}_{U_2}(P) = \mathsf{nf}_{U_2}(Q)$ and $\mathsf{nf}_{U_3}(P) = \mathsf{nf}_{U_3}(Q)$ then $P^n = Q^n$.*

Proof. We prove it by induction on n. If $n = 0$ then $P^0 = \perp = Q^0$. Suppose now that $n > 0$. The proof proceeds by cases.

1. Case $P = xP_1 \ldots P_k$. Since P and Q have the same nf_{U_2},

   $$\mathsf{nf}_{U_2}(Q) = \mathsf{nf}_{U_2}(P) = x\ \mathsf{nf}_{U_2}(P_1) \ldots \mathsf{nf}_{U_2}(P_k)$$

   The term Q being in $\beta\perp_{\mathcal{R}}$-normal form can $\beta\perp_{\mathcal{R}}$-reduce to a head normal form only if it is a head normal form itself. Hence, we have that $Q = xQ_1 \ldots Q_k$. Since P and Q have the same nf_{U_2} and the same nf_{U_3}, so do P_i and Q_i for all $1 \leq i \leq k$. Suppose $n > k$. Then,

   $$\begin{aligned} P^n &= xP_1^{n-k} \ldots P_k^{n-1} \\ &= xQ_1^{n-k} \ldots Q_k^{n-1} \text{ by induction hypothesis} \\ &= Q^n \end{aligned}$$

   Suppose $n \leq k$. Let $i = k - n$. Then,

   $$\begin{aligned} P^n &= \perp P_i^0 \ldots P_k^{n-1} \\ &= \perp Q_i^0 \ldots Q_k^{n-1} \text{ by induction hypothesis} \\ &= Q^n \end{aligned}$$

2. Case $P = \bot P_1 \dots P_k$. In this case, we have that $Q = \bot Q_1 \dots Q_k$ because U_2 does not contain any head active form. Then, we proceed as in the previous case.
3. Case $P = \lambda x.P_0$. In this case, we have that $Q = \lambda x.Q_0$ because U_2 does not contain any abstraction. P_0 and Q_0 have the same nf_{U_2} and the same nf_{U_3}. Then, by induction hypothesis, $P_0^{n-1} = Q_0^{n-1}$. Hence, $P^n = \lambda x.P_0^{n-1} = \lambda x.Q_0^{n-1} = Q^n$.
4. Case $P = ((\dots)P_2)P_1$ is a strong infinite left spine. We have two cases:
   (a) Case $P = ((((^\omega\mathbf{KI})P_k)\dots)P_2)P_1$ for some $k \geq 0$. Since P and Q have the same nf_{U_2} and the same nf_{U_3},

$$\begin{aligned}\mathsf{nf}_{U_2}(Q) &= \mathsf{nf}_{U_2}(P) = \bot\ \mathsf{nf}_{U_2}(P_k)\dots\mathsf{nf}_{U_2}(P_1)\\ \mathsf{nf}_{U_3}(Q) &= \mathsf{nf}_{U_3}(P) = \bot\ \mathbf{I}\ \mathsf{nf}_{U_3}(P_k)\dots\mathsf{nf}_{U_3}(P_1)\end{aligned}$$

   This is possible only if $Q = (^\omega\mathbf{KI})Q_k \dots Q_1$. Since P and Q have the same nf_{U_2} and the same nf_{U_3}, so do P_i and Q_i for all $1 \leq i \leq k$. Suppose $n \leq k$. Then,

$$\begin{aligned}P^n &= \bot P_n^0 \dots P_1^{n-1}\\ &= \bot Q_n^0 \dots Q_1^{n-1} \text{ by induction hypothesis}\\ &= Q^n\end{aligned}$$

   Suppose $n > k$. Then,

$$\begin{aligned}P^n &= (\mathbf{KI})^{n-k} P_k^{n-k} \dots P_1^{n-1}\\ &= (\mathbf{KI})^{n-k} Q_k^{n-k} \dots Q_1^{n-1} \text{ by induction hypothesis}\\ &= Q^n\end{aligned}$$

   (b) Otherwise, P is not of the form $((((^\omega\mathbf{KI})P_k)\dots)P_2)P_1$ for any $k \geq 0$. In this case, we have that $Q = ((\dots)Q_2)Q_1$ is also a strong infinite left spine because U_2 does not contain P. Since P and Q have the same nf_{U_2} and the same nf_{U_3}, so do P_i and Q_i for all $1 \leq i \leq k$. Then,

$$\begin{aligned}P^n &= \bot P_n^0 \dots P_1^{n-1}\\ &= \bot Q_n^0 \dots Q_1^{n-1} \text{ by induction hypothesis}\\ &= Q^n\end{aligned}$$

□

Theorem 40. *Let U_2 and U_3 be the sets of meaningless terms used in the proof of Theorem 37. We have that $\mathsf{nf}_{\mathcal{R}}(M) = \mathsf{nf}_{\mathcal{R}}(N)$ if and only if $\mathsf{nf}_{U_2}(M) = \mathsf{nf}_{U_2}(N)$ and $\mathsf{nf}_{U_3}(M) = \mathsf{nf}_{U_3}(N)$.*

Proof. (⇒) Suppose $\mathsf{nf}_{\mathcal{R}}(M) = \mathsf{nf}_{\mathcal{R}}(N)$. Then,

$$\mathsf{nf}_{U_2}(M) = \mathsf{nf}_{U_2}(\mathsf{nf}_{\mathcal{R}}(M)) = \mathsf{nf}_{U_2}(\mathsf{nf}_{\mathcal{R}}(N)) = \mathsf{nf}_{U_2}(N)$$

by Corollary 38 and because $\mathcal{R} \subseteq U_2$. Similarly, $\mathsf{nf}_{U_3}(M) = \mathsf{nf}_{U_3}(N)$.

(⇐) Suppose $\mathsf{nf}_{U_2}(M) = \mathsf{nf}_{U_2}(N)$ and $\mathsf{nf}_{U_3}(M) = \mathsf{nf}_{U_3}(N)$. Let $P = \mathsf{nf}_{\mathcal{R}}(M) = \mathsf{BerT}(M)$ and $Q = \mathsf{nf}_{\mathcal{R}}(N) = \mathsf{BerT}(N)$ (see Definition 7). By Corollary 38 and the fact that $\mathcal{R} \subseteq U_2, U_3$, we have that

$$\mathsf{nf}_{U_2}(P) = \mathsf{nf}_{U_2}(M) = \mathsf{nf}_{U_2}(N) = \mathsf{nf}_{U_2}(Q) \text{ and}$$
$$\mathsf{nf}_{U_3}(P) = \mathsf{nf}_{U_3}(M) = \mathsf{nf}_{U_3}(N) = \mathsf{nf}_{U_3}(Q).$$

By Lemma 39, $P^n = Q^n$ for all n. Hence, $P = Q$. □

Corollary 41. $\mathfrak{T}_{U_2} \cap \mathfrak{T}_{U_3} = \mathfrak{T}_{\mathcal{R}}$.

Theorem 42. *Let U_3 and U_4 be the sets of meaningless terms used in the proof of Theorem 37. We have that $\mathsf{nf}_{\mathcal{R}}(M) = \mathsf{nf}_{\mathcal{R}}(N)$ if and only if $\mathsf{nf}_{U_3}(M) = \mathsf{nf}_{U_3}(N)$ and $\mathsf{nf}_{U_4}(M) = \mathsf{nf}_{U_4}(N)$.*

The previous theorem is proved similarly to Theorem 41.

Corollary 43. $\mathfrak{T}_{U_3} \cap \mathfrak{T}_{U_4} = \mathfrak{T}_{\mathcal{R}}$

The next result is also proved in [12] using a different counterexample.

Theorem 44. *The lattice of lambda theories is neither modular nor distributive.*

Proof. The lattice of λ-theories contains the following sublattice isomorphic to N_5.

$$\begin{tikzcd}
 & \mathfrak{T}_5 = \mathfrak{T}_{\mathcal{R}} + \{\mathbf{\Omega} = \mathbf{Fix}(\lambda x.x\mathbf{K}),\ \mathbf{\Omega} = \mathbf{\Omega I},\ \mathbf{\Omega} = \mathbf{\Omega K}\} \arrow[dl] \arrow[ddr] & \\
\mathfrak{T}_4 = \mathfrak{T}_{\mathcal{R}} + \{\mathbf{\Omega} = (\mathbf{Fix}(\lambda x.x\mathbf{K}))\mathbf{I},\ \mathbf{\Omega} = \mathbf{\Omega I}\} \arrow[dd] & & \\
 & & \mathfrak{T}_3 = \mathfrak{T}_{\mathcal{R}} + \{\mathbf{\Omega} = \mathbf{Fix}(\lambda x.x\mathbf{K}),\ \mathbf{\Omega} = \mathbf{\Omega K}\} \arrow[ddl] \\
\mathfrak{T}_2 = \mathfrak{T}_{\mathcal{R}} + \{\mathbf{\Omega} = (\mathbf{Fix}(\lambda x.x\mathbf{K}))\mathbf{I}\} \arrow[dr] & & \\
 & \mathfrak{T}_1 = \mathfrak{T}_{\mathcal{R}} &
\end{tikzcd}$$

Note that the infinite normal form of $\mathbf{Fix}(\lambda x.x\mathbf{K})$ is ${}^{\omega}\mathbf{K}$ and the infinite normal form of $(\mathbf{Fix}(\lambda x.x\mathbf{K}))\mathbf{I}$ is ${}^{\omega}\mathbf{KI}$.

We have that $\{\mathfrak{T}_i \mid 1 \leq i \leq 5\}$ are all consistent because for all $1 \leq i \leq 5$, $\mathfrak{T}_i \subseteq \mathfrak{T}_{U_i}$ and $\mathfrak{T}_{U_i}$ is consistent by Corollary 38.

To prove that the above five theories form a sublattice of the lattice of λ-theories, we have to prove that it is closed under the join and meet operations, i.e. $\mathfrak{T}_5 = \mathfrak{T}_3 \sqcup \mathfrak{T}_4 = \mathfrak{T}_2 \sqcup \mathfrak{T}_3$ and $\mathfrak{T}_1 = \mathfrak{T}_2 \sqcap \mathfrak{T}_3$.

We first prove that $\mathfrak{T}_5 = \mathfrak{T}_3 \sqcup \mathfrak{T}_4$. It is clear that $\mathfrak{T}_3, \mathfrak{T}_4 \subseteq \mathfrak{T}_5$. For any $\mathfrak{T}$ such that $\mathfrak{T}_3, \mathfrak{T}_4 \subseteq \mathfrak{T}$, it is not difficult to prove that $\mathfrak{T}_5 \vdash M = N$ implies $\mathfrak{T} \vdash M = N$ by induction on the derivation. The derivation rules are the ones of Definition 2.1.4 of [1] extended to include the axioms of $\mathfrak{T}_{\mathcal{R}}$, $\mathbf{\Omega} = \mathbf{Fix}(\lambda x.x\mathbf{K})$, $\mathbf{\Omega} = \mathbf{\Omega I}$ and $\mathbf{\Omega} = \mathbf{\Omega K}$. Hence, $\mathfrak{T}_5 \subseteq \mathfrak{T}$ and $\mathfrak{T}_5$ is the smallest theory that contains $\mathfrak{T}_3$ and $\mathfrak{T}_4$. The equality $\mathfrak{T}_5 = \mathfrak{T}_2 \sqcup \mathfrak{T}_4$ is proved similarly.

We now prove that $\mathfrak{T}_1 = \mathfrak{T}_2 \sqcap \mathfrak{T}_3$, i.e. $\mathfrak{T}_1 = \mathfrak{T}_2 \cap \mathfrak{T}_3$. It is clear that $\mathfrak{T}_1 \subseteq \mathfrak{T}_2$ and $\mathfrak{T}_1 \subseteq \mathfrak{T}_3$. Hence, $\mathfrak{T}_1 \subseteq \mathfrak{T}_2 \cap \mathfrak{T}_3$. On the other hand, we have that $\mathfrak{T}_1 \cap \mathfrak{T}_2 \subseteq \mathfrak{T}_{U_1} \cap \mathfrak{T}_{U_2} = \mathcal{R}$ by Corollary 41. The proof of the equality $\mathfrak{T}_1 = \mathfrak{T}_3 \sqcap \mathfrak{T}_4$ is similar using Corollary 43. □

7 Conclusions

In spite of the fact that the interval $[\mathcal{R}, \Lambda^\infty]$ of all sets of meaningless terms cannot be decomposed as union of elementary key intervals (because of $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$), the problem of studying the whole lattice can be reduced to the problem of studying only three intervals: $[\mathcal{R}, \mathcal{SA} \cup \mathcal{SIL}]$, $[\mathcal{HA}, \mathcal{HA} \cup \mathcal{IL}]$ and $[\mathcal{HA} \cup \mathcal{O}, \mathcal{HA} \cup \mathcal{IL} \cup \mathcal{O}]$. We plan to investigate further what happens in these three intervals. There are far more sets of meaningless terms in these three intervals than the ones shown in this paper. The set $\{RM_1 \ldots M_{2n} \mid R \in \mathcal{R}, M_{2i} = \mathbf{I}$ and $M_{2i+1} = \mathbf{K}\}$ is a simple example of a meaningless set in $[\mathcal{R}, \mathcal{SA}]$ which is not of the form $\mathcal{SA}_X$ for any X. And we plan to study the relation between the lattice of meaningless sets and the lattice of lambda theories [11].

Acknowledgements. We would like to thank the reviewers for their detailed and helpful comments and suggestions that they provided.

References

1. Barendregt, H.P.: The Lambda Calculus: Its Syntax and Semantics. Revised edn. North-Holland, Amsterdam (1984)
2. Barr, M.: Terminal coalgebras for endofunctors on sets. Theoretical Computer Science 114(2), 299–315 (1999)
3. Berarducci, A.: Infinite λ-calculus and non-sensible models. In: Logic and Algebra (Pontignano, 1994), pp. 339–377. Dekker, New York (1996)
4. Davey, B.A., Priestly, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge (1990)
5. Kennaway, J.R., de Vries, F.J.: Infinitary rewriting. In: Terese (ed.) Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science, vol. 55, pp. 668–711. Cambridge University Press, Cambridge (2003)
6. Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.J.: Infinite lambda calculus and Böhm models. In: Hsiang, J. (ed.) RTA 1995. LNCS, vol. 914, pp. 257–270. Springer, Heidelberg (1995)
7. Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.J.: Transfinite reductions in orthogonal term rewriting systems. Information and Computation 119(1), 18–38 (1995)
8. Kennaway, J.R., Klop, J.W., Sleep, M.R., de Vries, F.J.: Infinitary lambda calculus. Theoretical Computer Science 175(1), 93–125 (1997)
9. Kennaway, R., Severi, P., Sleep, R., de Vries, F.-J.: Infinitary rewriting: From syntax to semantics. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F., de Vrijer, R. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity. LNCS, vol. 3838, pp. 148–172. Springer, Heidelberg (2005)
10. Kennaway, J.R., van Oostrom, V., de Vries, F.J.: Meaningless terms in rewriting. Journal of Functional and Logic Programming 1999(1) (February 1999)
11. Lusin, S., Salibra, A.: The lattice of lambda theories. Journal of Logic and Computation 14(3), 373–394 (2004)
12. Salibra, A.: Nonmodularity results for lambda calculus. Fundamenta Informaticae 45, 379–392 (2001)

13. Severi, P., de Vries, F.-J.: An extensional böhm model. In: Tison, S. (ed.) RTA 2002. LNCS, vol. 2378, pp. 159–173. Springer, Heidelberg (2002)
14. Severi, P., de Vries, F.J.: Continuity and discontinuity in lambda calculus. In: Urzyczyn, P. (ed.) TLCA 2005. LNCS, vol. 3461, pp. 369–385. Springer, Heidelberg (2005)
15. Severi, P., de Vries, F.-J.: Order structures on böhm-like models. In: Ong, L. (ed.) CSL 2005. LNCS, vol. 3634, pp. 103–118. Springer, Heidelberg (2005)
16. Severi, P., Vries, F.J.d.: A Lambda Calculus for D_∞. Technical report, University of Leicester (2002)

A Some Basic Lemmas

Lemma 45. *Let $U \subseteq \Lambda^\infty$ satisfy closure under substitution and β-reduction. If $xP_1 \dots P_n \in U$ then $U = \Lambda^\infty$.*

Proof. $M = xP_1 \dots P_n \in U$. Let $N \in \Lambda^\infty$ be arbitrary and $z_1 \dots z_k$ variables which are not in N. By the closure under substitution and reduction axioms, $M[x := \lambda z_1 \dots z_n.N] \twoheadrightarrow_\beta N \in U$. □

Lemma 46. *Let U be a meaningless set.*

1. *If $\lambda x.M \in U$ then $M \in U$.*
2. *If $\lambda x.M \in U$ then $\mathcal{HA} \subseteq U$. In particular, if $\mathbf{O} \in U$ then $\mathcal{HA} \subset U$.*
3. *If $\lambda x.M \in U$ then U is closed under abstractions, i.e. for all $P \in U$, we have that $\lambda x.P \in U$.*

Proof. 1. By the overlap and closure under β-reduction axioms, $(\lambda x.M)x \to_\beta M \in U$.

2. By the overlap axiom, $(\lambda x.M)Q \in U$ for all $Q \in \Lambda^\infty$. By indiscernibility we have that $RQ \in U$ for $R \in \mathcal{R}$ and also $RQ_1 \dots Q_k \in U$ for all $Q_i \in \Lambda^\infty$. By the previous part and indiscernibility, $\lambda x.R \in U$ and hence $\lambda x_1 \dots x_n.RQ_1 \dots Q_k \in U$.
3. If $\lambda x.M \in U$ then $M \in U$. By indiscernibility, $\lambda x.P \in U$ for any $P \in U$. □

Lemma 47. *1. If $\mathcal{SIL} \subseteq U$ then $\mathcal{SA} \subseteq U$. Hence, the minimal meaningless set containing $\mathcal{SIL}$ is $\mathcal{SA} \cup \mathcal{SIL}$.*

2. If $\mathcal{IL} \subseteq U$ then $\mathcal{HA} \subseteq U$. Hence, The minimal meaningless set containing $\mathcal{IL}$ is $\mathcal{HA} \cup \mathcal{IL}$.

Proof. 1. Let ${}^\omega Q = ((\dots)Q)Q)$. We have that ${}^\omega Q = {}^\omega QQ \in U$ By indiscernibility, $RQ \in U$ for any $R \in \mathcal{R}$ and also $RQ_1 \dots Q_k \in U$ for all $Q_i \in \Lambda^\infty$.

2. $\lambda x_1 \dots x_n.{}^\omega PQ_1 \dots Q_k \in U$. By indiscernibility, $\lambda x_1 \dots x_n.RQ_1 \dots Q_k \in U$. □

As a consequence of the previous lemma, there is no meaningless set between $\mathcal{R} \cup \mathcal{IL}$ and $\mathcal{HA} \cup \mathcal{IL}$ (hence, there is no point in including $\mathcal{R} \cup \mathcal{IL}$ as a key vertex).

B Checking That a Set is Meaningless

In this section we show the proof of Theorem 33 part 1. The rest of the theorems about meaningless sets are proved similarly. We give a criterion for proving that a set is meaningless where indiscernibility has to be checked only on terms whose common structure is a Berarducci tree. Checking this condition on some restricted set of terms will be enough provided the set U is closed under β-expansions. This criterion differs from the one in [15] on the fact that the common structure is now a Berarducci tree and not a skeleton.

Definition 48. Let $U \subseteq \Lambda^{\infty}$, $P, M, N \in \Lambda^{\infty}_{\bot}$.

1. $P \preceq_U M$ if P is obtained from M by replacing some subterms of M which belong to U by $\bot$.
2. We say that P is a common structure for M and N relative to U if $P \preceq_U M$ and $P \preceq_U N$.

E.g. $\bot\bot$ is a common structure for $\mathbf{\Omega\Omega}$ and $\mathbf{\Omega}(\mathbf{\Omega\Omega})$ with respect to $\mathcal{SA}$.

Definition 49. *[15]* The *skeleton* of a term $M \in \Lambda^{\infty}_{\bot}$ is defined by coinduction.

$$\begin{array}{ll}
\mathsf{skel}(M) = y & \text{if } M \twoheadrightarrow_{\beta} y \\
\mathsf{skel}(M) = \bot & \text{if } M \twoheadrightarrow_{\beta} \bot \\
\mathsf{skel}(M) = \lambda x.\mathsf{skel}(N) & \text{if } M \twoheadrightarrow_{\beta} \lambda x.N \\
\mathsf{skel}(M) = \mathsf{skel}(N)\ \mathsf{skel}(P) & \text{if } M \twoheadrightarrow_{\beta} NP \text{ and there is no } Q \text{ such that } N \twoheadrightarrow_{\beta} \lambda x.Q \\
\mathsf{skel}(M) = M & \text{if } M \text{ does not have a top normal form}
\end{array}$$

The skeleton of a term is essentially the Berarducci tree of a term but instead of replacing rootactive terms by $\bot$, we leave rootactive terms untouched.

Lemma 50. *Let $M \in \Lambda^{\infty}_{\bot}$. Then $M \twoheadrightarrow_{\beta} \mathsf{skel}(M)$ and $\mathsf{skel}(M)$ is either a head normal form, $\bot P_1 \dots P_k$, a head active form, an infinite left spine or $\mathbf{O}$.*

Note that $\mathsf{BerT}(M) = \mathsf{BerT}(\mathsf{skel}(M)) \preceq_U \mathsf{skel}(M)$ for any set $U \supseteq \mathcal{R}$.

Lemma 51. *Let U be closed under substitution. If $M \preceq_U N$ and $M \twoheadrightarrow_{\beta} M'$ then $N \twoheadrightarrow_{\beta} N'$ and $M' \preceq_U N'$ for some N'.*

Proof. This is proved by induction on the length of the reduction sequence. □

If the set U contains abstractions, from $M \preceq_U N$ and $N \twoheadrightarrow_{\beta_h} N'$, we may not be able to find M' such that $M \twoheadrightarrow_{\beta_h} M'$ and $M' \preceq_U N'$. For example, suppose U contains $\lambda x.\mathbf{\Omega}$. Then, $\bot\mathbf{I} \preceq_U (\lambda x.\mathbf{\Omega})\mathbf{I} \twoheadrightarrow_{\beta_h} \mathbf{\Omega}$ but $\bot\mathbf{I}$ cannot be obtained from $\mathbf{\Omega}$ by replacing terms in U by $\bot$.

Lemma 52. *Let U be closed under substitution and β-reduction. If $M \preceq_U N$ and $N \twoheadrightarrow_{\beta_h} N'$, then we have two cases:*

1. *$M \twoheadrightarrow_{\beta_h} M'$ and $M' \preceq_U N'$ for some M',*
2. *$M \twoheadrightarrow_{\beta_h} \lambda x_1 \dots x_k.\bot Q_1 \dots Q_n$ with $n \geq 1$ and U contains some abstraction.*

Proof. We prove it for one step of β_h-reduction. Suppose

$$N = \lambda x_1 \ldots x_k.(\lambda x.Q_0)Q_1 \ldots Q_n \text{ and}$$
$$N' = \lambda x_1 \ldots x_k.Q_0[x := Q_1]Q_2 \ldots Q_n.$$

Then we have four cases:

1. $M = \lambda x_1 \ldots x_i.\bot$. Then $M \preceq_U N'$.
2. $M = \lambda x_1 \ldots x_k.\bot Q_1 \ldots Q_n$. This case is possible only if U contains the abstraction $\lambda x.Q_0 \in U$ which has been replaced by $\bot$.
3. $M = \lambda x_1 \ldots x_k.(\lambda x.Q'_0)Q'_1 \ldots Q'_n$ with $Q'_i \preceq_U Q_i$ for all $0 \leq i \leq n$. Then

$$\begin{aligned} M' = \lambda x_1 \ldots x_k.Q'_0[x := Q'_1]Q_1 \ldots Q'_n \preceq_U N' \\ = \lambda x_1 \ldots x_k.Q_0[x := Q_1]Q_1 \ldots Q_n \end{aligned}$$

   This is because U is closed under substitutions and we have that $Q'_0[x := Q_1] \preceq_U Q_0[x := Q_1]$.
4. $M = \bot Q'_i \ldots Q'_n$ for $2 \leq i \leq n$. In this case $(\lambda x.Q'_0)Q'_1 \ldots Q'_{i-1} \in U$ has been replaced by $\bot$. Since U is closed under β-reduction, $Q'_0[x := Q'_1] \ldots Q'_{i-1} \in U$ and hence $M \preceq_U Q'_0[x := Q'_1] \ldots Q'_n$. □

Lemma 53. *Let U be closed under substitution and β-reduction. If $M \preceq_U N$ and M rootactive then N is rootactive.*

Proof. Suppose N is not rootactive. Then $N \twoheadrightarrow_{\beta_h} N'$ and N' is a top normal form. By Lemma 52, we have two cases

1. either $M \twoheadrightarrow_{\beta_h} \lambda x_1 \ldots x_k.\bot Q_1 \ldots Q_n$. Since M is rootactive, this case is not possible.
2. or we have that there exists M' such that $M \twoheadrightarrow_{\beta_h} M'$ and $M' \preceq_U N'$. If N' is a head normal form or an abstraction, so is M'. Then, these cases are not possible because M is rootactive. Now, suppose that N' is an application of the form $N_1 N_2$ where N_1 does not reduce to an abstraction. Then $M' = M_1 M_2$ with $M_1 \preceq_U N_1$ and $M_2 \preceq_U N_2$. Suppose towards a contradiction that M_1 reduces an abstraction $\lambda x.M_0$. By Lemma 51, N_1 reduces to a term N_3 such that $\lambda x.M_0 \preceq_U N_3$. Then, N_3 should be an abstraction as well. □

Lemma 54. *Let U satisfy rootactiveness, be closed under substitution and β-reduction. Let P be a skeleton, i.e.* $\mathsf{skel}(P) = P$. *If $P \preceq_U M$ then* $\mathsf{BerT}(P) \preceq_U M$.

Proof. $\mathsf{BerT}(P)$ is obtained from P by replacing all the rootactive subterms of P by $\bot$. We prove that $(\mathsf{BerT}(P))^n \preceq_U M^n$ for all n where M^n denotes the truncation of M at depth n. Since $(\mathsf{BerT}(P))^n$ is finite, we can proceed by induction on the number of symbols of $(\mathsf{BerT}(P))^n$. We show only the case when $P = P_0 \ldots P_k$ and P_0 is rootactive. Then $M = M_0 \ldots M_k$ and $P_i \preceq_U M_i$ for $0 \leq i \leq k$. By Lemma 53, M_o is rootactive. Since M_0 does not contain $\bot$'s, we have that $\bot \preceq_U M_0$. By Induction Hypothesis, $(\mathsf{BerT}(P_i))^n \preceq_U M_i^n$ for $1 \leq i \leq k$. Hence, $(\mathsf{BerT}(P))^n \preceq_U M^n$. □

Definition 55. *Let $U \subseteq \Lambda^\infty$. We say that U satisfies the axiom of* weak indiscernibility *if for all $P \in \mathsf{BerT}(\Lambda^\infty_\perp)$ such that $P \preceq_U M$ and $P \preceq_U N$, we have that $M \in U$ if and only if $N \in U$.*

Theorem 56. *Suppose $U \subset \Lambda^\infty$ satisfies: closure under β-reduction, closure under β-expansion, closure under substitution, rootactiveness and weak indiscernibility. Then U satisfies indiscernibility. If in addition U satisfies overlap, then U is a meaningless set.*

Proof. We prove indiscernibility. Let $M \overset{U}{\leftrightarrow} N$. Then there exists P such that $P \preceq_U M$ and $P \preceq_U N$. By Lemma 50 and Lemma 51, we have that $\mathsf{skel}(P) \preceq_U M'$ and $\mathsf{skel}(P) \preceq_U N'$ for some M', N' such that $M \twoheadrightarrow_\beta M'$ and $N \twoheadrightarrow_\beta N'$. By Lemma 54, $\mathsf{BerT}(\mathsf{skel}(P)) \preceq_U M'$ and $\mathsf{BerT}(\mathsf{skel}(P)) \preceq_U N'$. By weak indiscernibility, $M' \in U$ if and only if $N' \in U$. Since U is closed under β-reduction and β-expansion, we have that $M \in U$ if and only if $N \in U$. □

The following lemma will be used in the next proof of Theorem 33 part 1.

Lemma 57. *Suppose U satisfies the first four axioms of meaningless. Let $P \in \mathsf{BerT}(\Lambda^\infty_\perp)$ and $P \preceq_U M$. If $M \twoheadrightarrow_\beta M'$ and M' does not contain any subterm in U then $P = M = M'$.*

Proof. This is proved by induction on the length of the reduction sequence $M \twoheadrightarrow_\beta M'$. We prove the case when the length is 1. Let $M = C[(\lambda x.M_0)M_1]$ and $M' = C[M_0[x := M_1]]$. Since M' does not have any subterm in U, we have that $M_0[x := M_1] \notin U$. By closure under substitutions, $M_0 \notin U$. By overlapping $(\lambda x.M_0)$ and $(\lambda x.M_0)M_1 \notin U$. Then P should contain a β-redex of the form $(\lambda x.P_0)P_1$ where $P_0 \preceq_U M_0$ and $P_1 \preceq_U M_1$. But this contradicts the fact that P is in $\beta\perp$-normal form. □

We now prove Theorem 33 part 1.

Proof. We apply Theorem 56. We have to prove weak indiscernibility for $U = \mathcal{HA} \cup \mathcal{S}_X$. Suppose $P \in \mathsf{BerT}(\Lambda^\infty_\perp)$ and $P \preceq_{\mathcal{HA}\cup\mathcal{S}_X} M, N$.

1. If P is either a head normal form or **O**, so are M and N. Hence, $M, N \notin \mathcal{HA} \cup \mathcal{S}_X$.
2. Suppose $P = \lambda x_1 \dots \lambda x_n.\perp P_1 \dots P_k$ is a head bottom form. Then,
$$M = \lambda x_1 \dots x_n.M_o M_1 \dots M_k \text{ and } N = \lambda x_1 \dots x_n.N_0 N_1 \dots N_k$$
where $M_o, N_0 \in \mathcal{HA} \cup \mathcal{S}_X$ and $P_i \preceq_{\mathcal{HA}\cup\mathcal{S}_X} M_i, N_i$ for $0 \le i \le k$. We have two options for N_0. If $N_0 \in \mathcal{HA}$ then $N \in \mathcal{HA}$. And if $N_0 \in \mathcal{S}_X$ then $N \in \mathcal{S}_X$.
3. Suppose $P = \lambda x_1 \dots x_n.((\dots)P_2)P_1$ is an infinite left spine. Then,
$$M = \lambda x_1 \dots x_n.((\dots)M_2)M_1 \text{ and } N = \lambda x_1 \dots x_n.((\dots)N_2)N_1$$
where $P_i \preceq_{\mathcal{HA}\cup\mathcal{S}_X} M_i, N_i$ for all i. If $M \in \mathcal{S}_X$ then there exists l such that for all $m \ge l$, M_m reduces to a Lévy Longo tree without $\perp$. Hence $\mathsf{LLT}(M_m)$ does not contain any subterm in $\mathcal{HA} \cup \mathcal{S}_X$. By Lemma 57, we have that $\mathsf{LLT}(M_m) = M_m = P_m$. Since P_m does not contain $\perp$'s, we also have that $M_m = P_m = N_m$. Then, $N \in \mathcal{S}_X$.

Strong Normalization and Confluence for Reflexive Combinatory Logic

Daniyar S. Shamkanov

Department of Math. Logic and the Theory of Algorithms,
Faculty of Mechanics and Mathematics,
Moscow State University, Moscow 119992, Russia

Abstract. Reflexive combinatory logic RCL was introduced by S. Artemov as an extension of typed combinatory logic $\mathsf{CL}_{\rightarrow}$ capable to internalize its own derivations of typing judgments. It was designed as a basic theoretical prototype of functional programming languages extended by this kind of self-referential capacity. However, its operational aspects remained to be clarified.

We study reductions for reflexive combinatory logic and prove strong normalization and confluence properties.

1 Introduction

Reflexive combinatory logic RCL [10,11] was introduced by S. Artemov as a typed combinatory system supplied with some special kind of self-referential capacities. RCL extends typed combinatory logic $\mathsf{CL}_{\rightarrow}$ by a new type constuctor $t : F$ with the intended interpretation *"t has type F"* (see section 2). Informally speaking, if the term t represents a computable functional of type F, then $t : F$ can be considered as the set of high level descriptions (programming codes) of t. Thus the expressive power of such new types provides the possibility to operate simultaneously with objects of different abstraction level: functions, high level programs, low level codes, etc..

In addition, reflexive combinatory logic RCL continues the line of Curry-Howard isomorphism and the internalization property of Artemov's logic of proofs LP: it is a typed combinatory system capable to represent its own derivations by its own typed terms. RCL contains implicative intuitionistic logic, so it represents intuitionistic derivations. It also contains typed combinatory logic $\mathsf{CL}_{\rightarrow}$, so RCL represents derivations of $\mathsf{CL}_{\rightarrow}$. And so on.

Roughly in the same way as Artemov's logic of proofs LP relates to modal logic S4 the calculi considered in this paper should be related to modal lambda calculi. Such systems and their applications have been intensively studied, e.g., in [1,2,3,4,5,6,7]. However, the precise relationship between these formalisms and RCL is a matter of further investigation.

In [11] , N. Krupski established that typability and type restoration for RCL can be done in polynomial time and that the derivability relation for RCL is decidable and *PSPACE*-complete.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 228–238, 2011.

In [8], some version of reflexive λ-calculus was considered. The corresponding reflexive lambda terms were strongly normalizable but not confluent.

In this note, we study a system of reductions for RCL and prove strong normalization and confluence properties.

In the first section, we define the system of RCL and give some informal explanations. Then we consider natural reductions for RCL and define the system RCL^+ proposed by V. Krupski. In final section we prove that expressions of RCL^+ are strongly normalazible and confluent.

2 Reflexive Combinatory Logic

Reflexive combinatory logic RCL below is a joint calculus of propositions (types) and proofs (well-formed terms). We assume a Church style rigid typing, that is every term carries a fixed type.

Following [11], the definition of RCL has two parts: inductive definitions of types and derivability. Now we introduce the language of RCL and the notion of derivability from hypotheses.

2.1 The Language of RCL: Types and Typed Terms

Definition 1. *Terms* and *formulas* of RCL are inductively defined as follows:

– Propositional variables $\{p_0, p_1, \ldots\}$ are formulas;
– If F, G are formulas and u is a term, then $F \to G$, $u : F$ are formulas;
– If F is a formula, then then term variables $\{x_0^F, x_1^F, \ldots\}$ and term constants k^F, s^F, d^F, o^F, c^F are terms;
– If F is a formula and u, v are terms, then $(u \cdot v)^F$, $(!u)^F$ are terms.

Terms and formulas are called *expressions.*

We assume the connective : to bind stronger than $\to$. Thus, $u : F \to G$ is an abbreviation for $(u : F) \to G$.

For an expression e, all expressions that occur inside e including the occurrences inside superscripts, inside superscripts in superscripts, etc. are called *subexpressions* of e. A subexpression is called a *subterm* or a *subformula* when it is a term or a formula respectively.

Unlike the case of simply typed systems in RCL not all formulas are types.

Definition 2. The following inference system defines the notion of *type* (*well-formed formula*) of RCL:

$$(i)\ p_i \qquad (ii)\ \frac{F, G}{F \to G} \qquad (iii)\ \frac{F}{x_i^F : F}$$

$$(iv)\ \frac{u : (F \to G),\ v : F}{(uv)^G : G} \qquad (v)\ \frac{F, G}{k^{F \to (G \to F)} : (F \to (G \to F))}$$

$$(vi)\ \frac{F, G, H}{s^{(F\to(G\to H))\to((F\to G)\to(F\to H))} : ((F \to (G \to H)) \to ((F \to G) \to (F \to H)))}$$

$$(vii)\frac{u : F}{(!u)^{u:F} : (u : F)} \qquad (viii)\ \frac{u : F}{d^{u:F\to F} : (u : F \to F)}$$

$$(ix)\ \frac{u : (F \to G),\ v : F}{o^{u:(F\to G)\to(v:F\to(uv)^G:G)} : (u : (F \to G) \to (v : F \to (uv)^G : G))}$$

$$(x)\ \frac{u : F}{c^{u:F\to(!u)^{u:F}:(u:F)} : (u : F \to (!u)^{u:F} : (u : F))}$$

The first six rules came from $\mathsf{CL}_{\to}$; their meaning didn't change. Rule (vii) means that any type of the form $u : F$ has a canonical element $!u$ which, informally speaking, represents the high level description of u or the term u supplied with additional metadata (for instance, this can be data about types for all subterms of u). The remaining three rules introduce new combinators. The intended interpretation is the following: the combinator $d^{u:F\to F}$ maps the high level description of an object u into u itself, $o^{u:(F\to G)\to(v:F\to uv:G)}$ implements application on high level descriptions (or terms with metadata) and $c^{u:F\to!u:u:F}$ maps the description into the higher description.

Definition 3. A term u is called *typed* (*well-formed*) if there is a type of the form $u : F$ for some F.

Proposition 1 (N. Krupski [11]). *1. If v is a subterm of some well-formed formula then v is well-formed. Moreover, the formula G such that $v : G$ is well-formed is unique and also well-formed.*

2. Every subformula of a well-formed formula is well-formed.

2.2 Derivability Relation

The notion of derivability from hypotheses is defined as usual by axiom schemes and rule modus ponens.

Definition 4. *Axioms* of RCL are well-formed formulas of the following form:

1. $t : F \to F$;
2. $k^{(\dots)} : (F \to (G \to F))$;
3. $s^{(\dots)} : ((F \to (G \to H)) \to ((F \to G) \to (F \to H)))$;
4. $d^{(\dots)} : (u : F \to F)$;
5. $o^{(\dots)} : (u : (F \to G) \to (v : F \to (uv)^G : G))$;
6. $c^{(\dots)} : (u : F \to (!u)^{u:F} : (u : F))$.

Rule modus ponens: $F, F \to G \vdash G$.

All hypotheses must be well-formed. Modus ponens preserves well-formdness (follows from Proposition 1), so every formula in a valid derivation is well-formed.

RCL contains implicative intuitionistic logic, ordinary combinatory logic $\mathsf{CL}_{\to}$, and is closed under the combinatory application rule:

$$\frac{u:(F\to G),\ v:F}{(uv)^G:G}.$$

Furthermore, RCL enjoys the internalization property:

Proposition 2 (S. Artemov [10]). *If $A_1,\dots,A_n\vdash B$ then for any set of variables $x_1,\dots,x_n$ of respective types, it is possible to construct a term $t(x_1,\dots,x_n)$ such that*

$$x_1:A_1,\dots,x_n:A_n\vdash t(x_1,\dots,x_n):B.$$

3 Contractions

In this section we discuss natural contraction schemes, which express the intended meaning of corresponding combinators, and define the extended system RCL^+.

Consider contractions of the following form:

$$kuv\mapsto u,\quad suvw\mapsto(uw)(vw),\quad d!u\mapsto u,\quad c!u\mapsto !!u,\quad o(!u)(!v)\mapsto !(uv)$$

Although the schemes are very natural, the unrestricted contracting may transform a well-formed expression into the illegal one. As an example, consider a type $!(\mathtt{kxy}):(kxy:p_2)$, which can be reduced to $!\mathtt{x}:(kxy:p_2)$ but the latter is not a type according to our definition.

The first idea is to contract all redexes of the given contraction simultaneously.

Definition 5. The application of a contraction $a\mapsto b$ to an expression e means simultaneous replacement of all occurrences of a in e by b. The resulting expression will be denoted by $e[a\mapsto b]$.

This definition secures well-formedness preservation under the contractions for expressions without the combinator o^F. In particular,

$$o^{kx:(p_1\to p_2)\to(y:p_1\to \mathtt{kxy}:p_2)}(!kx)(!y)$$

has two normal forms. It can be reduced to $!((kx)y)$ and then to $!x$. Another possibility is an illegal term

$$o^{kx:(p_1\to p_2)\to(y:p_1\to \mathtt{x}:p_2)}(!kx)(!y);$$

the reduction was made in the upper index.

To provide the preservation for all expressions, V. Krupski proposed to extend RCL to RCL^+ with the following conditions:

— if o^F is a combinator and $a\mapsto b$ is a contraction, then $o^{F[a\mapsto b]}$ is also a combinator,

— if $a \mapsto b$ and $c \mapsto d$ are contractions and a is not syntactically equal to c, then $a[c \mapsto d] \mapsto b[c \mapsto d]$ is also a contraction.

The first condition provides well-formedness preservation for all expressions. The second one provides confluence.

Theorem 1. *The extended system* RCL^+ *has strong normalization and confluence properties.*

Now we present RCL^+ and its basic properties more formally. The theorem will be proved in the next section.

Definition 6. The set of *well-formed formulas* of RCL^+ is defined by rules (i)-(x) of RCL with additional rule:

$$(xi)\ \frac{o^F : F}{(o^F : F)[a \mapsto b]}, \quad \text{if } a \mapsto b \text{ is a contraction of } \mathsf{RCL}^+.$$

Definition 7. A term u is called *well-formed* (according to RCL^+) if there is a well-formed formula of RCL^+ of the form $u : F$ for some F.

Definition 8. All redexes and contracta below must be well-formed. *Contractions* of RCL^+ are the following:

$$kuv \mapsto u, \quad suvw \mapsto (uw)(vw), \quad d!u \mapsto u, \quad c!u \mapsto !!u, \quad o(!u)(!v) \mapsto !(uv),$$

$$(R)\frac{a \mapsto b, \quad c \mapsto d}{a[c \mapsto d] \mapsto b[c \mapsto d]}, \quad \text{if } a \text{ is not syntactically equal to } c.$$

By Greek letters γ, δ, etc. we denote finite (possibly empty) sequences of contractions. A successive application of a sequence $\gamma = a_0 \mapsto b_0, \dots, a_n \mapsto b_n$ to an expression e is $e\gamma = e[a_0 \mapsto b_0] \dots [a_n \mapsto b_n]$.

Definition 9. Let $a \mapsto b$ be a contraction of RCL^+. The *rank* $rk(a \mapsto b)$ is the number of applications of rule R in the shortest derivation of $a \mapsto b$ plus one. For a sequence of contractions γ, let $rk(\gamma)$ be the sum of ranks of its corresponding contractions.

It easily follows that each contraction $a \mapsto b$ of RCL^+ has one of the following forms:

(k) $kuv \mapsto u$;
(s) $s(u\tau)(v\tau)(w\tau) \mapsto ((uw)(vw))\tau$, where $rk(a \mapsto b) > rk(\tau)$;
(d) $d!u \mapsto u$;
(o) $o!(u\tau)!(v\tau) \mapsto (!(uv))\tau$, where $rk(a \mapsto b) > rk(\tau)$;
(c) $c!u \mapsto !!u$;

A contraction of corresponding form is called *k-, s- ,d-, o-,* or *c- contraction*, respectively.

Now we give precise definition of the reduction relation and establish basic closure properties of RCL^+. We adopt standard notation for binary relations: $\to^*$ means reflexive and transitive closure of $\to$, $\to^+$ means transitive closure.

Definition 10. An expression e *one-step reduces* to e' (notation $e \to_{\mathsf{RCL}^+} e'$) if e' is not syntactically equal to e and has the form $e[a \mapsto b]$ for some contraction $a \mapsto b$ of RCL^+. As usual, we say e *reduces* to e' (notation $e \to^*_{\mathsf{RCL}^+} e'$) if and only if e' is obtained from e by a finite (possibly empty) series of one-step reductions.

Lemma 1. *If both formulas $u : F$ and $u : G$ are well-formed, then F and G coincide.*

Proof. It can easily be checked by induction on derivations of well-formed formulas that if $v^H : J$ is well-formed, then H coincides with J. The term u has the form v^H for some H. So H, F and G coincide. □

Lemma 2. *If F is well-formed formula and $F \to_{\mathsf{RCL}^+} F'$, then F' is also well-formed.*

Proof. It is sufficient to show that if H is well-formed formula and $a \mapsto b$ is a contraction of RCL^+, then $H[a \mapsto b]$ is well-formed. The prove is by induction on derivations of well-formedness of H.

Consider the case of rule (iv):

$$\frac{u : (F \to G),\ v : F}{(uv)^G : G}.$$

H has the form $(uv)^G : G$. Suppose a coincides with $(uv)^G$. So a doesn't occur in G, and $G[a \mapsto b] \equiv G$. So we have $H[a \mapsto b] \equiv b : G$. By definition of contractions, the term b is well-formed and has the same superscript as a, i.e. b has the form w^G. Hence $b : G$ is a well-formed.

If $a \not\equiv (uv)^G$. Then $H[a \mapsto b] \equiv (u[a \mapsto b]v[a \mapsto b])^{G[a \mapsto b]}$. And by the inductive assumption for $u[a \mapsto b] : (F[a \mapsto b] \to G[a \mapsto b])$ and $v[a \mapsto b] : F[a \mapsto b]$, $H[a \mapsto b]$ is well-formed.

All other cases are trivial. □

Lemma 3. *Every subexpression of a well-formed formula is well-formed.*

Proof. For every contraction $a \mapsto b$, the terms a and b are well-formed. Thus we can modify rule (xi) as follows:

$$\frac{o^F : F,\ b : G}{(o^F : F)[a \mapsto b]}, \quad \text{if } a \mapsto b \text{ is a contraction of } \mathsf{RCL}^+.$$

Evidently, the set of well-formed expressions remains the same.

Now we show that every subexpression of a well-formed formula F is well-formed. The proof is by induction on the derivation of well-formedness of F in the modified inference system.

Consider the case of modified rule. Suppose e is a subexpression of $(o^F : F)[a \mapsto b]$. If there exists a subexpression f of $o^F : F$ such that $f[a \mapsto b] \equiv e$, then f is well-formed by inductive assumption for $o^F : F$, and $f[a \mapsto b]$ is well-formed by Lemma 2. Otherwise e is a subexpression of b, and e is well-formed by inductive assumption for $b : G$.

Cases of other rules are trivial. □

Theorem 2. *1. If v is a subterm of some well-formed formula then v is well-formed. Moreover, the formula G such that $v : G$ is well-formed is unique and also well-formed.*

2. Every subformula of a well-formed formula is well-formed.

3. If e is well-formed and $e \to_{\mathsf{RCL}^+} e'$, then e' is also well-formed.

4 Strong Normalization and Confluence

In this section we prove strong normalization and confluence for RCL^+. For this purpose, we study some supplementary systems of expressions and reductions first.

4.1 Modal Combinatory Logic

We consider a variant of typed combinatory logic which corresponds to intuitionistic modal logic $\mathsf{IS4}$ (cf. [1], [4], [2]). We denote this system by $\mathsf{CL}_{\to\Box}$.

Definition 11. Types and typed terms of $\mathsf{CL}_{\to\Box}$ are

Types:

$$p_i, \qquad \frac{F, G}{F \to G}, \qquad \frac{F}{\Box F};$$

Typed terms:

$$x_i^F, \qquad \frac{u^{F\to G},\ v^F}{(uv)^G}, \qquad \frac{u^F}{(!u)^{\Box F}},$$

$$k^{F\to(G\to F)}, \qquad s^{(F\to(G\to H))\to((F\to G)\to(F\to H))},$$
$$o^{\Box(F\to G)\to(\Box F\to\Box G)}, \qquad d^{\Box F\to F}, \qquad c^{\Box F\to\Box\Box F}.$$

Definition 12. The one-step reduction ($\to_{\mathsf{CL}_{\to\Box}}$) is defined by the following contraction schemes:

$$kuv \mapsto_{\mathsf{CL}_{\to\Box}} u, \quad suvw \mapsto_{\mathsf{CL}_{\to\Box}} (uw)(vw),$$

$$d!u \mapsto_{\mathsf{CL}_{\to\Box}} u, \quad o(!u)(!v) \mapsto_{\mathsf{CL}_{\to\Box}} !(uv), \quad c!u \mapsto_{\mathsf{CL}_{\to\Box}} !!u.$$

More precisely,

1. The terms of the forms kuv, $suvw$, $d!u$, $o(!u)(!v)$, $c!u$ are called redexes, with u, $(uw)(vw)$, u, $!(uv)$, $!!u$ being their contracta, respectively.

2. If term u contains an occurrence of a redex r and we replace that occurrence by its contractum, and the resulting term is u', we say u one-step reduces to u' (notation $u \to_{\mathsf{CL}_{\to\Box}} u'$).

3. We say u reduces to u' (or u computes to u', notation $u \to^*_{\mathsf{CL}_{\to\Box}} u'$) if and only if u' is obtained from u by a finite (possibly empty) series of contractions.

Theorem 3. *The system* $\mathsf{CL}_{\to\Box}$ *has strong normalization and confluence properties.*

Proof. First note that the given system of contractions does not have critical pairs, so it is locally confluent.

Now let us define an interpretation of $\mathsf{CL}_{\to\Box}$ into simply typed lambda calculus: for a type F we omit boxes in it and denote the result by F°; for typed terms let

$$(x_i^F)^\circ = x_i^{F^\circ}, \quad ((uv)^G)^\circ = (u^\circ v^\circ)^{G^\circ}, \quad ((!u)^F)^\circ = u^\circ,$$

$$(k^F)^\circ = (\lambda xy.x)^{F^\circ}, \quad (s^F)^\circ = (\lambda xyz.xz(yz))^{F^\circ},$$

$$(d^F)^\circ = (\lambda x.x)^{F^\circ}, \quad (o^F)^\circ = (\lambda xy.xy)^{F^\circ}, \quad (c^F)^\circ = (\lambda x.x)^{F^\circ}.$$

Obviously, if $u \to_{\mathsf{CL}_{\to\Box}} v$, then $u^\circ \to^+_{\lambda_\to} v^\circ$ (here $\to^+_{\lambda_\to}$ is the transitive closure of one-step reduction relation of $\lambda_\to$); so all typed terms of $\mathsf{CL}_{\to\Box}$ are strongly normalizible. According to Newman's lemma, local confluence and strong normalization imply confluence. □

4.2 Adequate Expressions and Reductions

Now we study a system of adequate expressions and reductions. For an expression e, let all subformulas of the form $t : F$ be substituted by $\Box F$. The result of the procedure will be denoted by $e^\bullet$.

Definition 13. An expression of RCL (any expression, not necessary a type or a typed term) is called adequate if, for any its subexpression e, $e^\bullet$ is a type or typed term of $\mathsf{CL}_{\to\Box}$.

Notice that all well-formed expressions of RCL^+ are adequate.

In the definition below, we don't require well-formedness of redexes or contracta. Thus we write out all superscripts.

Definition 14. The one-step reduction ($\rightsquigarrow$) is defined by the following contraction schemes:

1. $((k^{F\to G\to F}u^F)^{G\to F}v^G)^F \Rightarrow u^F$,
2. $(((s^{(F\to(G\to H))\to((F\to G)\to(F\to H))}u^{F\to(G\to H)})^{(F\to G)\to(F\to H)}v^{F\to G})^{F\to H}w^F)^H \Rightarrow ((u^{F\to(G\to H)}w^F)^{G\to H}(v^{F\to G}w^F)^G)^H$,
3. $(d^{u^F:F\to F}(!u^F)^{u^F:F})^F \Rightarrow u^F$,
4. $(((o^{u^{F\to G}:(F\to G)\to(v^F:F\to \mathbf{w}^G:G)}(!u^{F\to G})^{u^{F\to G}:(F\to G)})^{(v^F:F\to\mathbf{w}^G:G)})(!v^F)^{v^F:F})^{\mathbf{w}^G:G}$ $\Rightarrow (!(u^{F\to G}v^F)^G)^{\mathbf{w}^G:G}$,
5. $(c^{u^F:F\to(!u^F)^{u^F:F}:u^F:F}(!u^F)^{u^F:F})^{(!u^F)^{u^F:F}:u^F:F} \Rightarrow (!(!u^F)^{u^F:F})^{(!u^F)^{u^F:F}:u^F:F}$.

All contraction schemes have precisely the same form as before except that, in the upper indexes of the forth scheme, uv is substituted by $\mathbf{w}$. Evidently, if redexes of schemes are adequate, then contracta are adequate too.

Theorem 4. *The reduction relation $\rightsquigarrow$ on the set of adequate expressions possesses strong normalization and confluence properties.*

Proof. As in the case of $\mathsf{CL}_{\to\Box}$, the given system of contractions has no critical pairs. Thus it is locally confluent. Confluence will be stated when we'll prove strong normalization.

For an expression e, we denote the maximal number of nested superscript levels in e by $h(e)$. Note that if $e \rightsquigarrow e_1$, then $h(e_1) \leqslant h(e)$.

We say that $e \rightsquigarrow e_1$ is a *zero-level reduction* if the corresponding redex occurs in e not inside a superscript. If $v \rightsquigarrow v_1$ is a zero-level reduction, then $v^\bullet \to_{\mathsf{CL}_{\to\Box}} v_1^\bullet$.

We prove strong normalization for e by induction on $h(e)$. Suppose e is not strongly normalizible; then there is an infinite series of reductions $e \rightsquigarrow e_1 \rightsquigarrow e_2 \rightsquigarrow \ldots$. If there are only finitely many zero-level reductions in the series, then we can find e_n and a superscript F from e_n such that infinite number of reductions occur in F. But $h(F) < h(e_n) \leqslant h(e)$; by inductive hypotheses, F is strongly normalazible. This contradiction means that there are infinite number of zero-level reductions in the series. Thus we can find a subterm u from e such that infinitely many zero-level reductions occur in u. Consequently, there is an infinite series of reductions $u^\bullet \to_{\mathsf{CL}_{\to\Box}} \ldots$. But $u^\bullet$ is a typed terms of $\mathsf{CL}_{\to\Box}$; hence it is strongly normalizible.

This contradiction concludes the proof. □

4.3 Final Considerations

Lemma 4. *Suppose $(uv)^G$ is a well-typed term of RCL^+ and γ is a finite sequence of contractions of RCL^+; then $(u\gamma v\gamma)^{G\gamma} \rightsquigarrow^* (uv)^G\gamma$.*

Proof. The proof is by induction on $rk(\gamma)$. If $rk(\gamma) = 0$, then γ is empty and there is nothing to prove. Suppose $rk(\gamma) > 0$. Let δ be a seqence of contractions such that $\gamma = a \mapsto b,\ \delta$.

If $(u[a \mapsto b]v[a \mapsto b])^{G[a\mapsto b]} \equiv (uv)^G[a \mapsto b]$, then by the inductive assumption for δ we have

$$(u\gamma v\gamma)^{G\gamma} \equiv (u[a \mapsto b]\delta v[a \mapsto b]\delta)^{G[a\mapsto b]\delta}$$

$$\rightsquigarrow^* (u[a \mapsto b]v[a \mapsto b])^{G[a\mapsto b]}\delta \equiv (uv)^G[a \mapsto b]\delta \equiv (uv)^G\gamma.$$

Assume the converse. Then we get $a \equiv (uv)^G$ and $(u[a \mapsto b]\delta v[a \mapsto b]\delta)^{G[a\mapsto b]\delta} \equiv (u\delta v\delta)^{G\delta}$. The contraction $a \mapsto b$ is k-, s-, d-, o- or c- contraction.

Case 1. $a \mapsto b$ has the form $kmn \mapsto m$. Then we have

$$(u\delta v\delta)^{G\delta} \equiv ((km\delta)(n\delta))^{G\delta} \equiv$$
$$\equiv (k(m\delta)(n\delta))^{G\delta} \rightsquigarrow m\delta \equiv (kmn)^G[kmn \mapsto m]\delta \equiv (uv)^G[a \mapsto b]\delta.$$

Case 2. $a \mapsto b$ is $(s(l\theta)(m\theta)(n\theta))^{F\theta} \mapsto ((ln)(mn))\theta$ where θ is a sequence of contractions such that $rk(a \mapsto b) > rk(\theta)$. By the inductive assumption for $\theta\delta$ we have

$$(u\delta v\delta)^{G\delta} \equiv ((s(l\theta)(m\theta))\delta(n\theta\delta))^{F\theta\delta} \equiv$$
$$\equiv (s(l\theta\delta)(m\theta\delta)(n\theta\delta))^{F\theta\delta} \rightsquigarrow (l\theta\delta n\theta\delta)(m\theta\delta n\theta\delta) \rightsquigarrow^* (ln)\theta\delta(mn)\theta\delta \rightsquigarrow^*$$
$$((ln)(mn))\theta\delta \equiv (uv)^G[(s(l\theta)(m\theta)(n\theta))^{F\theta} \mapsto ((ln)(mn))\theta]\delta \equiv (uv)^G[a \mapsto b]\delta.$$

Case 3. $a \mapsto b$ has the form $d!m \mapsto m$. We have

$$(u\delta v\delta)^{G\delta} \equiv (d\delta(!m)\delta)^{G\delta} \equiv$$
$$\equiv (d!(m\delta))^{G\delta} \rightsquigarrow m\delta \equiv (d!m)^G[d!m \mapsto m]\delta \equiv (uv)^G[a \mapsto b]\delta.$$

Case 4. $a \mapsto b$ is $(o!(m\theta)!(n\theta))^{F\theta} \mapsto !((mn)\theta)$, where $rk(a \mapsto b) > rk(\theta)$. By the inductive assumption for $\theta\delta$ we have

$$(u\delta v\delta)^{G\delta} \equiv ((o!(m\theta))\delta(!(n\theta))\delta)^{F\theta\delta} \equiv$$
$$\equiv (o\delta(!(m\theta))\delta(!(n\theta))\delta)^{F\theta\delta} \equiv (o!(m\theta\delta)!(n\theta\delta))^{F\theta\delta} \rightsquigarrow$$
$$!(m\theta\delta n\theta\delta) \rightsquigarrow^* !((mn)\theta\delta) \equiv (!((mn)\theta))\delta \equiv$$
$$\equiv (o!(m\theta)!(n\theta))^G[o!(m\theta)!(n\theta) \mapsto !((mn)\theta)]\delta \equiv (uv)^G[a \mapsto b]\delta.$$

Case 5. $a \mapsto b$ has the form $c!m \mapsto !!m$.

$$(u\delta v\delta)^{G\delta} \equiv (c\delta(!m)\delta)^{G\delta} \equiv$$
$$\equiv (c!(m\delta))^{G\delta} \rightsquigarrow !!(m\delta) \equiv (!!m)\delta \equiv (c!m)^G[c!m \mapsto !!m]\delta \equiv (uv)^G[a \mapsto b]\delta.$$

All cases are done. □

Lemma 5. $e_1 \rightarrow_{\mathsf{RCL}^+} e_2 \Longrightarrow e_1 \rightsquigarrow^+ e_2$.

Proof. It is sufficient to prove the following: if $a \mapsto b$, then $a \rightsquigarrow^+ b$. The cases of k-, d- and c-reductions are obvious.

Let $a \mapsto b$ be of the form $(s(l\delta)(m\delta)(n\delta))^{F\delta} \mapsto ((ln)(mn))\delta$. We have

$$(s(l\delta)(m\delta)(n\delta))^{F\delta} \rightsquigarrow (l\delta n\delta)(m\delta n\delta).$$

The applications of the previous lemma for (ln) and δ, for (mn) and δ, and for $(ln)(mn)$ and δ yield

$$(l\delta n\delta)(m\delta n\delta) \rightsquigarrow^* (ln)\delta(mn)\delta \rightsquigarrow^* ((ln)(mn))\delta.$$

Thus we obtain $(s(l\delta)(m\delta)(n\delta))^{F\delta} \rightsquigarrow^+ ((ln)(mn))\delta$.

Let $a \mapsto b$ be of the form $(o!(m\delta)!(n\delta))^{F\delta} \mapsto !((mn)\delta)$. By the previous lemma we have

$$(o!(m\delta)!(n\delta))^{F\delta} \rightsquigarrow !(m\delta n\delta) \rightsquigarrow^* !((mn)\delta).$$

□

Lemma 6. *All normal forms of* RCL^+ *are also* $\rightsquigarrow$*-normal forms.*

Proof. Every $\rightsquigarrow$-redex of well-formed expression of RCL^+ is also a redex of RCL^+. □

Proof (Proof of Theorem 1). Strong normalization follows from Lemma 5 and strong normalization for adeqate reductions.

Suppose there is a well-formed expression e of RCL^+ which is not confluent. So it has at least two different normal forms e_1 and e_2. Moreover, e_1 and e_2 are normal forms with respect to adequate reductions. So the adeqate expression e has at least two $\rightsquigarrow$-normal forms. This contradicts Theorem 4. □

References

1. Bierman, G.M., de Paiva, V.C.V.: On an Intuitionistic Modal Logic. Studia Logica 65(3), 383–416 (2000)
2. Pfenning, F., Wong, H.C.: On a modal lambda-calculus for S4. In: Brookes, S., Main, M. (eds.) Proc. of the 11th Conference on Mathematical Foundations of Programming Semantics. Electronic Notes in Theoretical Computer Science, vol. 1, pp. 515–534. Elsevier, Amsterdam (1995)
3. Davies, R., Pfenning, F.: A modal analysis of staged computation. In: Steele Jr., G. (ed.) Proc. of the 23rd Annual Symposium on Principles of Programming Languages, pp. 258–270. ACM Press, New York (1996)
4. Martini, S., Masini, A.: A computational interpretation of modal proofs. In: Wansing, H. (ed.) Proof Theory of Modal Logics, pp. 213–241. Kluwer, Dordrecht (1996)
5. Wickline, P., Lee, P., Pfenning, F., Davies, R.: Modal types as staging specifications for run-time code generation. ACM Computing Surveys 30(3es), article 8 (1998)
6. Goubault-Larrecq, J., Goubault, E.: On the geometry of Intuitionistic S4 proofs. Homotopy, Homology and its Applications 5(2), 137–209 (2003)
7. Murphy VII, T., Crary, K., Harper, R., Pfenning, F.: A symmetric modal lambda calculus for distributed computing. In: Ganzinger, H. (ed.) Proc. of LICS 2004, pp. 286–295. IEEE Computer Society Press, Los Alamitos (2004)
8. Alt, J., Artemov, S.: Reflective λ-calculus. In: Kahle, R., Schroeder-Heister, P., Stärk, R.F. (eds.) PTCS 2001. LNCS, vol. 2183, pp. 22–37. Springer, Heidelberg (2001)
9. Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
10. Artemov, S.: Kolmogorov and Gödel's approach to intuitionistic logic: current developments. Russian Mathematical Surveys 59(2), 203–229 (2004)
11. Krupski, N.: Typing in Reflective Combinatory Logic. Annals of Pure and Applied Logic 141(1-2), 243–256 (2006)

On Polymorphic Types of Untyped Terms

Rick Statman

Carnegie Mellon University,
Department of Mathematical Sciences,
Pittsburgh, PA 15213
statman@cs.cmu.edu

Abstract. Let \$ be a finite set of beta normal closed terms and M and N a pair of beta normal, eta distinct, closed terms. Then there exist polymorphic types a, b such that every member of \$ can be typed as a, and M and N have eta expansions which can be typed as b ; where, in the resulting typings, the members of \$ can be simultaneously consistently identified, and the eta expansions of M and N are beta-eta inconsistent (no model with more than one element of any type). A similar result holds in the presence of surjective pairing.

Keywords: lambda calculus, lambda calculus with surjective pairing, polymorphic typings, Böhm's theorem.

1 Introduction

Alonzo Church believed that only beta normal untyped lambda terms have meaning and Corrado Böhm showed that no two such eta distinct terms can be identified in the untyped calculus. In addition, any beta normal term has a polymorphic type (I learned this from John Reynolds some 25 years ago but I am not sure of the origin), so the question arises as to the consistency of identification after various polymorphic typings. Indeed, Corrado had already asked me, during a visit to Roma, whether his theorem is true in various typed contexts. Below we shall prove a very strong result.

We work in the untyped lambda calculus, but we will type certain terms with polymorphic types (from Girard's system F_0 [5]). We shall also consider the untyped lambda calculus with surjective pairing. In this case we will type untyped terms with polymorphic types which we will assume are closed under pairing. This is somewhat stronger than having products but somewhat weaker than assuming surjective pairing on the typed side.

Polymorphic types $a, b, c, \ldots$ are built up from type variables $p, q, r, \ldots$ by $\rightarrow$ and $\bigwedge$ (under the Curry-Howard isomorphism, the universal quantifier). When typing an untyped term we require that each term variable have a fixed type and that all subterms are simultaneously compatibly typed (sometimes referred to as "Church typing"). In addition, when typing an untyped term, the type operations of abstraction ($\lambda p,\ \lambda q,\ \lambda r, \ldots$) corresponding to $\bigwedge$, and application can be inserted before and after symbols in the term. We shall prove the following theorem which we state first for the case without pairing.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 239–256, 2011.

Let \$ be a finite set of beta normal closed terms and M and N a pair of beta normal, eta distinct, closed terms. Then there exist polymorphic types a, b such that every member of \$ can be typed as a, and M and N have eta expansions which can be typed as b; where, in the resulting typings, the members of \$ can be simultaneously consistently identified, and the eta expansions of M and N are beta-eta inconsistent (no model with more than one element of any type).

Here, all the members of \$ can be identified in a non-trivial beta-eta model of the polymorphic lambda calculus with no empty types.

2 Untyped Terms

We begin with some preliminaries for the untyped calculus with pairing. The atoms of the language consist of

- variables; $x, y, z, \ldots$
- and constants; P, L, R
- Terms of the language are defined recursively;
- atoms are terms and
- if X, Y are terms then so are (XY) and λxX

We shall adopt the customary conventions;

- parens are deleted and restored by left association and the use of Church's infixed "dot" notation
- parens are added around abstractions, and additional unary operations for readability.

The axiom and rules of untyped lambda calculus are the following.
The first 5 axioms correspond to the classical theory of untyped lambda calculus with surjective pairing SP.

$$\begin{array}{lrl}
\text{(beta)} & (\lambda xX)\ Y = [Y/x]X & \\
\text{(eta)} & X = \lambda x.\ Xx & (x \text{ not free in } X) \\
(L/Pa) & L(PXY) = X & \\
(R/Pa) & R(PXY) = Y & \\
(P/Dp) & P(LX)(RX) = X &
\end{array}$$

The next 6 axioms correspond to the extended theory of Stovering (FP) and Statman (PSP, in the combinator case), which enjoys the Church Rosser property when formulated by reductions.

$$\begin{array}{lr}
(Ap/P) & PXYZ = P(XZ)(YZ) \\
(L/Ap) & LXY = L(XY) \\
(R/Ap) & RXY = R(XY) \\
(L/Ab) & L(\lambda xX) = \lambda x(LX) \\
(R/Ab) & R(\lambda xX) = \lambda x(RX) \\
(P/Ab) & P(\lambda xX)(\lambda xY) = \lambda x(PXY)
\end{array}$$

There are certain useful derived rules.

(1) (P/Dp) and $(Ap/P) \Rightarrow (L/Ap)$ and (R/Ap)

$$L(XY) = L(P(LX)(RX)Y) = L(P(LXY)(RXY)) = LXY \ ;$$

similarly for R.

(2) (L/Ap) and (R/Ap) and $(P/Dp) \Rightarrow (Ap/P)$

$$\begin{aligned} &L(PXYZ) = L(PXY)Z = XZ \qquad \text{and} \\ &R(PXYZ) = R(PXY)Z = YZ \end{aligned}$$

therefore,

$$PXYZ = P(L(PXYZ))(R(PXYZ)) = P(XZ)(YZ) \ .$$

(3) (eta) and $(P/Ap) \Rightarrow (P/Ab)$

$$\begin{aligned} P(\lambda xX)(\lambda xY) = \lambda y.\ P(\lambda xX)(\lambda xY)y = \\ \lambda y.\ P((\lambda xX)y)((\lambda XY)y) = \lambda x\ PXY \ . \end{aligned}$$

(4) (eta) and $(L/Ap) \Rightarrow (L/Ab)$

$$L(\lambda xX) = \lambda y.L(\lambda xX)y = \lambda y.L((\lambda xX)y) = \lambda x(LX) \ ;$$

similarly for R.

(5) (eta) $\Rightarrow LP = K$ and $RP = K^*$

$$L(PX) = \lambda x.\ L(PX)x = \lambda x.L(PXx) = \lambda x.X$$

thus,

$$LP = \lambda x.\ LPx = \lambda x.L(Px) = \lambda xy.x$$

similarly,

$$R(PX) = \lambda x.R(PX)x = \lambda x.\ R(PXx) = \lambda x.x$$

hence,

$$Rp = \lambda yx.x \ .$$

The result of Klop is that the Church-Rosser property fails for the following classical reductions for SP;

$$\begin{array}{lrl} \text{(beta)} & (\lambda xX)Y \to [Y/x]X & \\ \text{(eta)} & \lambda x.\ Xx \to X & (x \text{ not free in } X) \\ (L/Pa) & L(PXY) \to X & \\ (R/Pa) & R(PXY) \to Y & \\ (P/Dp) & P(LX)(RX) \to X & \end{array}$$

Nevertheless, this theory was proved conservative over beta-eta by de Vrijer. Stovering and, later, Statman (for the combinator case) introduced new reductions for the first 5 and additional reductions corresponding to the latter 6 which enjoy Church-Rosser.

Stovering Reductions for FP with eta

(beta)	$(\lambda x X)\ Y \to [Y/x]X$	
(etae)	$X \to \lambda x.\ Xx$	(x not free in X)
(L/Pa)	$L(PXY) \to X$	
(R/Pa)	$R(PXY) \to Y$	
(P/De)	$X \to P(LX)(RX)$	
(Ap/P)	$PXYZ \to P(XZ)(YZ)$	

Stovering-Statman Reductions for FP-PSP without eta

(beta)	$(\lambda x X)Y \to [Y/x]X$
(L/Pa)	$L(PXY) \to X$
$R/Pa)$	$R(PXY) \to Y$
(P/De)	$X \to P(LX)(RX)$
(Ap/P)	$PXYZ \to P(XZ)(YZ)$
(L/Ab)	$L(\lambda x X) \to \lambda x(LX)$
(R/Ab)	$R(\lambda x X) \to \lambda x(RX)$
(P/Ab)	$P(\lambda x X)(\lambda x Y) \to \lambda x\ PXY$

Statman Reductions for PSP without eta

(Ap/L)	$LXY \to L(XY)$
(Ap/R)	$RXY \to R(XY)$

Now, Church-Rosser is enough to obtain de Vrijer's theorem and a co-de Vrijer theorem that beta-eta is conservative over pairing with FP. Nevertheless, there are no normal forms in the presence of (etae) and (P/De). Indeed, every term FP reduces to one without any beta redexes

$$(\lambda x X)Y \twoheadrightarrow P(L(\lambda x X)Y)(R(\lambda x X)Y)\ .$$

So, another notion is needed. The theory of weak pointwise surjective pairing is defined by the following axioms.

$WPSP$

$$\begin{array}{rrl}
\text{(etae)} & X = \lambda x.Xx & (x \text{ not free in } X) \\
(L/Pa) & L(PXY) = X & \\
(R/Pa) & R(PXY) = Y & \\
(P/Dp) & P(LX)(RX) = X & \\
(L/Ap) & LXY = L(XY) & \\
(R/Ap) & RXY = R(XY) & \\
(L/Ab) & L(\lambda xX) = \lambda x(LX) & \\
(R/Ab) & R(\lambda xX) = \lambda x(RX) & \\
(P/Ab) & P(\lambda xX)(\lambda xY) = \lambda xPXY &
\end{array}$$

Definition 1. X *is* well proportioned *if*

- *$X = PXY$ and X and Y are well proportioned*
- *$X = \lambda x(1)\ldots x(r).D(1)(\ldots(D(d)(x\ \ X(1)\ldots X(s)))\ldots)$, where $X(j)$ are well proportioned and the $D(i)$ are L or R. Here d, r, or s can be 0.*

Lemma 1. *The lambda abstraction of a well proportioned term = a well proportioned term in $WPSP$.*

Proof. By induction on the definition.

Lemma 2. *Prefixing L or R to a well proportioned term = a well proportioned term in $WPSP$.*

Proof. By induction on the definition.

Lemma 3. *Well proportioned terms are closed under classical* (P/Dp) *reduction.*

Lemma 4. *If X contains no* (beta), (L/Pa), (R/Pa), (Ap/P), (Ap/L), *and* (Ap/R) *redexes, then X = a well proportioned term in $WPSP$.*

Proof. By induction on X. If X is an atom then we have $L = \lambda y.\ (Ly)$, $R = \lambda y.\ (Ry)$, and $P = P(\lambda yz.\ z)(\lambda y.\ y)$. For the induction step we distinguish several cases.

Case 1: X begins with lambda. Then apply the induction hypothesis and Lemma 1.

Case 2: X begins with L. Then since X has no (Ap/L) redexes X is LY and the induction hypothesis applies to Y. Now use Lemma 2.

Similarly, for R.

Case 3: X begins with P. Since X has no (Ap/P) redex we have

Subcase 1: X is PY. Then the induction hypothesis applies to Y and there is a well proportioned term $Z = Y$. Now $PY = P(\lambda x.\ Z)(\lambda x.\ x)$ and we can apply Lemma 1 to $\lambda x.\ Z$.

Subcase 2: X is PYZ. Apply the induction hypothesis to both Y and Z.

Because P, L, R occur in well proportioned terms only in very predictable ways we generalize the notion of the Böhm tree of a normal term to the "term tree of well proportioned term". The node corresponding to the subterm

$$\lambda x(1) \dots x(r).D(1)(\dots(D(d)(x\ X(1)\dots X(s)))\dots)$$

has label $\lambda x(1)\dots x(r) * D(1)\dots D(d) * x$ with s descendants and the node corresponding to the subterm PXY has label P and has two descendants.

3 Types

Types of the second-order calculus are defined recursively as follows:

- $p, q, r, \dots$ are type variables;
- if a is a type then so is $\bigwedge p\ a$, for any type variable p;
- if a and b are types then so is $(a \rightarrow b)$.

We shall employ Curry's substitution prefix both for terms and types. $[b/p]a$ is the result of substituting b for p in a with the usual provision for changing bound variables to avoid collision.

Typed terms are defined recursively as follows:

- $x\hat{\ }a, y\hat{\ }b, z\hat{\ }c, \dots$ are typed variables of types resp. $a, b, c, \dots$;
- if X and Y have type a then $\langle X, Y\rangle$ has type a, LX has type a, and RY has type a so $\langle , \rangle$ is a binary operation and L and R are unary operations which always occur in function position;
- if X has type b then $\lambda x\hat{\ }aX$ has type $a \rightarrow b$;
- if X has type $a \rightarrow b$ and Y has type a then (XY) has type b;
- if X has type a then $\lambda p\ X$ has type $\bigwedge p\ a$ for any type variable p;
- if X has type $\bigwedge p\ a$ and b is a type then $x\ b$ has type $[b/p]a$.

Here $\langle , \rangle$ is always used as a binary operation and there is one for each type a. Along with beta and eta conversion we only assume the analogue of (L/Pa) and (R/Pa)

$$L\langle X, Y\rangle \rightarrow X$$
$$R\langle X, Y\rangle \rightarrow Y\ .$$

The resulting theory, beta-eta-P, enjoys the Church-Rosser property and satisfies strong normalization. This can be proved by Girard's method. This is the theory for which we prove the second part of the result. For the first part it is convenient to consider a stronger theory beta-eta-PP. For what follows we often suppress type abstraction and application when it plays no role in the discussion.

$$\begin{array}{rll}
\text{(beta)} & (\lambda x X)Y \to [Y/x]X & \\
\text{(eta)} & \lambda x.\ Xx \to X & (x \text{ not free in } X) \\
(L/Pa) & L\langle X,Y\rangle \to X & \\
(R/Pa) & R\langle X,Y\rangle \to Y & \\
(Ap/P) & \langle X,Y\rangle Z \to \langle (XZ),(YZ)\rangle & \\
(L/Ap) & LXY \to L(XY) & \\
(R/Ap) & RXY \to R(XY) & \\
(L/Ab) & \lambda x(LX) \to L(\lambda x X) & \\
(R/Ab) & \lambda x(RX) \to R(\lambda x X) & \\
(P/Ab) & \lambda x\ \langle X,Y\rangle \to \langle (\lambda x X),(\lambda x Y)\rangle &
\end{array}$$

Using this theory also gives a stronger result. Strong normalization and Church-Rosser can be proved but do not fit in the space available in this volume. It will be useful here to consider the shape of beta-eta-PP normal forms.

Each normal term has the form of a binary tree of $\langle,\rangle$'s where each leaf consists of a sequence of applications of L's and R's to a lambda term in head normal form

$$\lambda x(1)\ldots x(r)xX(1)\ldots X(t)\ ,$$

where each $X(i)$ is of the same sort.

Since we have strong normalization and Church-Rosser, we have standard reductions to normal forms. However, the notion of head redex must be defined by depth first search:

- the head redex of $\lambda x(1)\ldots x(r).\ (\lambda x X)Y\ X(1)\ldots X(t)$ is $(\lambda x X)Y$;
- the head redex of $\lambda x(1)\ldots x(r).\ \langle X,Y\rangle\ X(1)\ldots X(t)$ is $\langle X,Y\rangle X(1)$ if $t>0$ and $\lambda x(r).\langle X,Y\rangle$ if $t=0$ but $r>0$;
- the head redex of $\lambda x(1)\ldots x(r).\ LX(1)\ldots X(t)$ is $L\ X(1)$ if $X(1)=\langle X,Y\rangle$, or otherwise $L\ X(1)\ X(2)$ if $t>1$, or else $\lambda x(r).\ L\ X(1)$ if $r>0$, or finally if all else fails the head redex of $X(1)$ and similarly for R.

The following simple remark will be useful below. If X is a term of type $\bigwedge p\ p$ containing at most free variables of type p and the constant C of type $\bigwedge p\ p$, then it is not possible for there to exist terms $X(1)\ldots X(t)$ with the same free variables and constant as X such that $X\ X(1)\ldots X(t)$ has type p and $X\ X(1)\ldots X(t) \twoheadrightarrow x^p$. For, if we suppose that X is normal it must have the form

$$\begin{array}{ll}
D(1)(\ldots(D(d)\ (C\ Y(1)\ldots Y(s)))\ldots) & \text{or} \\
D(1)(\ldots(D(d)(\lambda q.\ C\ Y(1)\ldots Y(s)))\ldots)\ . &
\end{array}$$

4 Typing Untyped Terms

Theorem 1. *Let $\$$ be finite set of well proportioned closed terms and M and N a pair of well proportioned closed terms not $WPSP$ equivalent. Then there exist polymorphic types a, b such that every member of $\$$ can be typed as a, and M and N have WPSP equivalents which can be typed as b; where, in the resulting typings, the members of $\$$ can be simultaneously consistently identified, and the $WPSP$ equivalents of M and N cannot be consistently identified.*

Remark 1. In the absence of pairing, $WPSP$ can be replace by eta equivalence. If pairing is not present in the untyped lambda calculus, it is not needed in typing, consistency, and inconsistency.

Proof. The first part of the proof concerns only $\$$ and the "folklore" polymorphic typing. For this each variable gets type $\bigwedge p\ p$. Now each subterm is typed recursively. Let t be the maximum length of a lambda prefix in any member of $\$$. If the subterm is

$$\lambda x(1) \dots x(r).D(1)(\dots(D(d)(x\ X(1) \dots X(s)))\dots)$$

and $X(i)$ is already typed $a =$

$$\bigwedge p\ p \to (\dots(\bigwedge p\ p \to \bigwedge p\ p)\dots)$$

for $t+1$ occurrences of $\bigwedge p\ p$ then type $c =$

$$a \to (\dots(a \to (\bigwedge p\ p \to (\dots(\bigwedge p\ p \to \bigwedge p\ p)\dots))\dots)$$

for s occurrences of a and $t - r + 1$ occurrences of $\bigwedge p\ p$ is inserted between x and $X(1)$ which results in a term of type a. If the subterm is PXY and both X and Y have type a already then PXY gets type a.

Now we must show that all the typed terms of $\$$ can be identified in a beta-eta-PP model of the polymorphic calculus with no empty types. We want no empty types just for a little stronger result, it is not an intrinsic condition. Toward this end we introduce a new constant C of type $\bigwedge p\ p$. We next construct the term model on this constant. There is at least one closed term Cc of each type (closed) type c. We wish to show that in this model it is not the case that $\lambda p\ \lambda x\hat{}\ p\ \lambda y\hat{}\ p\ x = \lambda p\ \lambda x\hat{}\ p\ \lambda y\hat{}\ p\ y$. Suppose to the contrary.

Then by Jacopini's theorem [6] there are closed terms $M(1), \dots M(m)$ and terms $P(1)$, $Q(1), \dots,$ $P(m)$, $Q(m)$ in $\$$ such that

$$\begin{aligned} x^p &= \text{beta} - \text{eta}{-}PP\ M(1)\ p\ x\hat{}\ p\ y\hat{}\ p\ C\ P(1)\ Q(1) \\ M(1)\ p\ x\hat{}\ p\ y\hat{}\ p\ C\ Q(1)\ P(1) &= \text{beta} - \text{eta}{-}PP\ M(2)\ p\ x\hat{}\ p\ y\hat{}\ p\ C\ P(2)\ Q(2) \\ \vdots \qquad\qquad & \qquad\qquad \vdots \\ M(m)\ p\ x\hat{}\ p\ y\hat{}\ p\ C\ Q(m)\ P(m) &= \text{beta} - \text{eta}{-}PP\ y\hat{}\ p\ . \end{aligned}$$

Now since x^p is normal,

$$M(1)\ p\ x^p y^p C\ P(1)\ Q(1) \twoheadrightarrow \text{beta} - \text{eta}{-}PP\ x^p\ .$$

W.l.o.g. we may assume that the $M(i)$ are beta-eta-PP normal. By standardization and eta-postponement (or strong normalization),

$$M(1)\ px\,\hat{}\,py\,\hat{}\,p\ C\ P(1)\ Q(1) \twoheadrightarrow x\,\hat{}\,p$$

by head reductions alone. Now $M(1)$ cannot begin with $\langle , \rangle$. Thus $M(1)$ has the form

$$D(1)\ (\ldots (D(d)\ \lambda x(1) \ldots x(r).(x\ X(1) \ldots X(s))) \ldots)$$

omitting type abstractions. Thus the head reduction goes to

$$D(1)\ (\ldots (D(d)((\lambda x(1) \ldots x(r).(x\ X(1) \ldots X(s)))px\,\hat{}\,py\,\hat{}\,p\ C\ P(1)\ Q(1)) \ldots)\ .$$

Since this must be of type p the prefix $\lambda x(1) \ldots x(r)$ must actually be

$$\lambda q\ \lambda u\,\hat{}\,q \lambda v\,\hat{}\,q \lambda y\,\hat{}\,a \lambda z\,\hat{}\,a\ .$$

So if $x = x\,\hat{}\,p$ and $s = 0$ then $d = 0$ and

$$M(1)\ px\,\hat{}\,py\,\hat{}\,p\ C\ P(1)\ Q(1) \twoheadrightarrow x\,\hat{}\,p\ .$$

The only other possibility is for $x = y$ or z. If we suppose that $x = y$ then the head reduction goes to

$$D(1)\ (\ldots (D(s)\ P(1) X'(1) \ldots X'(s)) \ldots)$$

with $X'(i) = [P(1)/y, Q(1)/z, x/u, y/v]X(i)$ and thus s is at least t and each $X'(i)$ has type $\bigwedge p\ p$ for i less than or equal to t. But this contradicts the remark at the end of the previous paragraph. Similarly, for $x = z$. Thus,

$$M(1)\ p\ x\,\hat{}\,py\,\hat{}\,p\ C\ Q(1)\ P(1) \twoheadrightarrow \text{beta} - \text{eta}{-}PP\ x\,\hat{}\,p\ .$$

Picking m as small as possible leads to a contradiction.

We now turn to the second part of the theorem. Suppose that we are given M and N with no (beta), $(L/Pa), (R/Pa), (Ap/P)$, (Ap/L), and (Ap/R) redexes. By Lemma 4 we may assume that they are well proportioned. We begin with some preparation of M and N.

Select t at least the maximum number of arguments of any variable occurrence in either M or N and the maximum number of consecutive P's along any path in M or N. We now (etae) and (P/De) reduce (classically expand) M and N by downward recursion on their term trees so that every occurrence of a variable, except those at the leaves of the resulting trees, has exactly $t + 2$ arguments and every maximal subterm beginning with P is a complete binary tree of depth t. These classical eta expansions will be referred to below as "ordinary". After (P/De) reductions additional (L/Ab) and (R/Ab) reductions may need to be applied to move new L's and R's to the right of λ's. This process should be carried out along each branch except those passing through one of the last two arguments so that if nodes corresponding to the last two arguments are omitted, the result is a complete t-ary tree of uniform depth h. Consequently, the results have the same unlabelled term tree. Moreover, h should be selected so that all the leaves result from (etae). It is convenient to continue to refer to the resulting eta expansions as M and N. These are certainly still well proportioned. Indeed, if we compare corresponding nodes of the term tree we either have both of the form PXY or

$$\lambda x(1)\ldots x(r)u\ v.\ D(1)\ (\ldots(D(d)(x\ X(1)\ldots X(t)u\ v))\ldots)\text{ in }M$$

and

$$\lambda y(1)\ldots y(s)u\ v.\ E(1)\ (\ldots(E(e)(y\ Y(1)\ldots Y(t)u\ v))\ldots)\text{ in }N\ ,$$

where the $D(i)$ and the $E(j)$ are either L or R, and we have made explicit the last two variables u, v which resulted from (etae) and occur nowhere in the $X(i)$ or $Y(j)$. We note that if these nodes differ there are three overlapping cases.

(i) $D(1)\ldots D(d) = / = E(1)\ldots E(e)$;
(ii) $r \neq s$;
(iii) $x \neq y$.

We now assume that there is some node at which they differ.

We now assign types to the subterms of M and N recursively upward from the leaves of the term trees. At each step certain further eta expansions will be introduced to ensure that the types of corresponding subterms, both between M and N and within M and N, with different length lambda prefixes, actually coincide. These eta expansions will be referred to as "extra-ordinary" expansions and they come in three sorts. The first sort can occur anywhere. The second sort consist of a single expansion following all ordinary ones and ones of the first sort, but these occur only at depths equal to or smaller than the first place at which the term trees of M and N differ. The third sort consist of $t+2$ expansions following the second sort of expansion but only at the depth equal to the first place at which the term trees of M and N differ. Additional (L/Ab) and (R/Ab) reductions may be needed after (etae) to put the untyped term back in well proportioned form.

The type assignment has several import properties.

(i) Every original, ordinary, and extra-ordinary eta variable has the same type $a =$

$$\bigwedge o \bigwedge p.\ \ o \to (\ldots(o \to (p \to (p \to \\ [\bigwedge q.\ \ (o \to (\ldots(o \to (p \to (p \to q)))\ldots)) \to q])))\ldots)\ .$$

(ii) The type of any extra-ordinary eta variable depends only on the depth of its ancestor in the term tree.
(iii) The type of any argument to P or any of the first t arguments to a variable occurrence depends only on its depth in the tree. Let f be the smallest depth

at which M and N differ. We begin the recursion; every original variable gets type a and this fixes the basis case. For the induction step we consider corresponding nodes of M and N at some depth k. We distinguish 3 cases depending on whether $k > f$, $k = f$, or $k < f$. Consider the corresponding nodes

$$\lambda x(1)\dots x(r)u\ v.\ D(1)(\dots(D(d)(x\ X(1)\dots X(t)u\ v))\dots) \text{ in M}$$
$$\lambda y(1)\dots y(s)u\ v.\ E(1)(\dots(E(e)(y\ Y(1)\dots Y(t)u\ v))\dots) \text{ in } N\ .$$

Compute non-negative integers m, n by first finding the maximum of the number l of lambda's in any prefix at depth k in either M or N and setting $m = l - r - 2$ and $n = l - s - 2$. W.l.o.g. we may assume s is at least r.

Case 1: $k > f$ We use the first sort of extra-ordinary eta expansions, followed by additional (L/Ab) and (R/Ab),

$$\lambda x(1)\dots x(r)u\ v\ z(1)\dots z(m)$$
$$D(1)(\dots(D(d)(x\ X(1)\dots X(t)u\ v\ z(1)\dots z(m)))\dots) \text{ in M}$$

$$\lambda y(1)\dots y(s)u\ v\ z(1)\dots z(n)$$
$$E(1)(\dots(E(e)(y\ Y(1)\dots Y(t)u\ v\ z(1)\dots z(n)))\dots) \text{ in N}$$

Case 2: $k < f$ We use the first and second sort of extra-ordinary eta expansions, followed by additional (L/Ab) and (R/Ab),

$$\lambda x(1)\dots x(r)u\ v\ z(1)\dots z(m)\lambda w$$
$$D(1)\dots(D(d)(x\ X(1)\dots X(t)u\ v\ z(1)\dots z(m)\ w))\dots) \text{ in M}$$

$$\lambda y(1)\dots y(s)u\ v\ z(1)\dots z(n)\lambda w$$
$$E(1)(\dots(E(e)\ (y\ Y(1)\dots Y(t)u\ v\ z(1)\dots z(n)\ w))\dots) \text{ in N}$$

Case 3: $k = f$ We use all three sorts of extra-ordinary eta expansions, followed by additional (L/Ab) and (R/Ab),

$$\lambda x(1)\dots x(r)u\ v\ z(1)\dots z(m)\lambda w\lambda w(1)\dots w(t+3)$$
$$D(1)(\dots(D(d)(x\ X(1)\dots X(t)u\ v\ z(1)\dots z(m)\ w\ w(1)\dots w(t+3)))\dots)\text{in M}$$

$$\lambda y(1)\dots y(s)u\ v\ z(1)\dots z(n)\lambda w\lambda w(1)\dots w(t+3)$$
$$E(1)(\dots(E(e)(y\ Y(1)\dots Y(t)u\ v\ z(1)\dots z(n)\ w\ w(1)\dots w(t+3)))\dots) \text{ in N}$$

We type as follows. In Case 1, we may assume that the subterms $X(i)$ and $Y(i)$ all have the same type b. Set

$$c = \bigwedge q.\ (b \to (\dots(b \to (a \to (a \to q))\dots)) \to q$$

Then we type apply x and y to $b\ a$. This results in $x\ X(1)\dots X(t)u\ v$ and $y\ Y(1)\dots Y(t)u\ v$ having type c. Now we type apply this term to c and each $z(i)$ to $b\ a$, and both

$$x\ X(1)\dots X(t)u\ v\ z(1)\dots z(i)$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(i)$$

to c. The end result has type c, and so the corresponding subterms in M and N have the same type

$$a \to (\ldots(a \to c)\ldots)$$

with l occurrences of a. In Case 3, we proceed as in Case 1 except

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)$$

are both type applied to b, w gets type $a' =$

$$\bigwedge o \bigwedge p(o \to (\ldots(o \to (p \to (p \to o)))\ldots))$$

and is type applied to $b\ a$ and the end result

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)w$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)w$$

has type b. So, the corresponding subterms in M and N have the same type

$$a \to (\ldots(a \to (a' \to b))\ldots)$$

with l occurrences of a. Now we consider Case 2. We proceed as in Case 3 except w gets type $a'' =$

$$\bigwedge o \bigwedge p(o \to (\ldots(o \to (p \to (p \to p)))\ldots))$$

and is type applied to $b\ a$, and

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)$$

are both type applied to a. So

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)w$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)w$$

are then of type a. Let $a''' = \bigwedge p\ p \to (p \to p)$. Then we give each $w(i)$ type a''' for $i < t+3$, we give $w(t+3)$ type

$$(a''' \to (\ldots(a''' \to (a''' \to (a''' \to a''')))))$$

for $t+3$ occurrences of a'''. If we type apply

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)w$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)w$$

to $a'''a'''$ and then

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)\ w\ (1)\ldots w(t+2)$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)\ w\ w(1)\ldots w(t+2)$$

to a''' this gives

$$x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)\ w\ w(1)\ldots w(t+3)$$

and

$$y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)\ w\ w(1)\ldots w(t+3)$$

type a'''. Thus, the corresponding subterms in M and N have the same type

$$\begin{aligned}&a \to (\ldots(a \to (a'' \to (a''' \to (\ldots(a''' \to\\ &\qquad\qquad ((a'' \to (a''' \to (\ldots(a''' \to a''')\ldots))) \to a'''))\ldots)))) \ldots)\end{aligned}$$

for l occurrences of a and $t+2$ initial occurrences of a'''. This completes the type assignment.

We are particularly interested in two terms of type a:

$$\begin{aligned}A = {}&\lambda o\ \lambda p(\lambda x(1)\hat{}\,o\ldots x(t)\hat{}\,o\ \lambda u\hat{}\,p\ \lambda v\hat{}\,p\\ &\lambda q\lambda w\hat{}\,(o \to (\ldots(o \to (p \to (p \to q)))\ldots))\ w\ x(1)\ldots x(t)\ u\ v\end{aligned}$$

$$\begin{aligned}B = {}&\lambda o\lambda p\ (\lambda x(1)\hat{}\,o\ldots x(t)\hat{}\,o\ \lambda u\hat{}\,p\ \lambda v\hat{}\,p\\ &\lambda q\ \lambda w\hat{}\,(o \to (\ldots(p \to (p \to (\to q)))\ldots))\ w\ x(1)\ldots x(t)\ v\ u\end{aligned}$$

We define the notion of a 1-pairing tree by

- $\lambda p\lambda x\hat{}\,p.\ x\hat{}\,p$ is a 1-pairing tree.
- If $T(1)$ and $T(2)$ are 1-pairing trees then $\lambda p\ \lambda x\hat{}\,p\langle T(1)px, T(2)px\rangle$ is a 1-pairing tree.

Each 1-pairing tree has type $\bigwedge p\ p \to p$.

2-pairing trees are defined by

- $\lambda p\ \lambda x\hat{}\,p\lambda y\hat{}\,p\ .\ x\hat{}\,p$ is a 2-pairing tree;
- $\lambda p\ \lambda x\hat{}\,p\lambda y\hat{}\,p.\ y\hat{}\,p$ is a 2-pairing tree;
- if $T(1)$ and $T(2)$ are 2-pairing trees then so is $\langle T(1), T(2)\rangle$ a 2-pairing tree.

2-pairing tree have type $\bigwedge p\ p \rightarrow (p \rightarrow p)$.

We shall now show that the typed versions of M and N are inconsistent with beta-eta conversion. That is, they have no model which both identifies them and has at least one type with more than one element. We shall proceed as in Böhm's theorem, [2], node by node down the term tree but the exact procedure depends on what happens at the uppermost leftmost position # where they differ, which by previous convention is at depth f. We recall the 3 cases

(i) $D(1)\ldots D(d) \neq E(1)\ldots E(e)$
(ii) $r \neq s$
(iii) $x \neq y$

Case (i)

W.l.o.g. we may assume that d is not larger than e.

We proceed down the term tree. At a node at depth smaller than f, if the position # lies within the ith descendent of the node then either the current node is labelled with P and we can apply L or R (the same on both the M and N side) to proceed downward or we have

$$\begin{aligned}&\lambda x(1)\ldots x(r)u\ v\ z(1)\ldots z(m)\ w\\ &D(1)(\ldots(D(d)(x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)\ w))\ldots) \text{ in M}\end{aligned}$$

$$\begin{aligned}&\lambda y(1)\ldots y(s)u\ v\ z(1)\ldots z(n)\ w\\ &D(1)(\ldots(D(d)(y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)\ w))\ldots) \text{ in N },\end{aligned}$$

where $X(i)$ and $Y(i)$ have the same type b. Let T be the 1-pairing tree which is complete binary of depth d. We apply both our terms to

$$A\ldots A(\lambda x(1)\hat{}\,b\ldots\lambda x(t)\hat{}\,b\ \lambda u\hat{}\,a\ \lambda v\hat{}\ \ a.\ T\ b\ x(i))$$

with l occurrences of A. The result of the Böhm-out on the M side is the result of substituting A for free variables in $X(i)$ and on the N side the result of substituting A for free variables in $Y(i)$. At depth f we have

$$\begin{aligned}&\lambda x(1)\ldots x(r)u\ v\ z(1)\ldots z(m)\lambda w\lambda w(1)\ldots w(t+3)\\ &D(1)(\ldots(D(d)\\ &\qquad(x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)\ w\ w\ (1)\ldots w(t+3)))\ldots) \text{ in M}\end{aligned}$$

$$\begin{aligned}&\lambda y(1)\ldots y(s)u\ v\ z(1)\ldots z(n)\lambda w\lambda w\ w(1)\ldots w(t+3)\\ &E(1)(\ldots(E(e)\\ &\qquad(y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)\ w\ w(1)\ldots w(t+3)))\ldots) \text{ in N}\end{aligned}$$

and we distinguish two subcases.

Subcase (a): There exists some $d' < d + 1$ such that

$$D(d') \neq E(d') \ .$$

Let T be a 2-pairing tree whose binary term tree has $K = \lambda p \lambda x\,\hat{}\,p\ y\,\hat{}\,p.\ x\,\hat{}\,p$ at the leaf whose binary position corresponds to the sequence $D(1) \ldots D(d)$ and $K^* = \lambda p \lambda x\,\hat{}\,p\ y\,\hat{}\,p$ at the leaf whose binary position corresponds to $E(1) \ldots E(e)$.

We apply both our terms to

$$A \ldots A\ K \ldots K\ (\lambda x(1)\,\hat{}\,a'' \ldots \lambda x(t)\,\hat{}\,a''\ \lambda u\,\hat{}\,a'''\ \lambda v\,\hat{}\,a'''\ .\ T)$$

for l occurrences of A and $t+2$ occurrences of K. The result of the Böhm-out on the M side is K and on the N side is K^*.

Subcase (b): $D(1) \ldots D(d)$ is a proper initial segment of $E(1) \ldots E(e)$.

We may write $E(1) \ldots E(e)$ as $D(1) \ldots D(d)\ F(1) \ldots F(g)$ where $g = e - d$. We proceed as in Subcase (a) by selecting a 2-pairing tree with $\langle K^*, K^* \rangle$ at the leaf corresponding to $E(1) \ldots E(e)$ and with K at a leaf corresponding to $D(1) \ldots D(d)\ G$ where G is the opposite of $F(1)$. The result of the Böhm-out on the M side is some term $\langle H, K \rangle$ or $\langle K, H \rangle$ and on the N side it is $\langle K^*, K^* \rangle$. So we can apply either L or R to get K on the M side and K^* on the N side.

Case (ii)

Case (ii) but not (i), so $r < s$. We proceed as in Case (i) until depth k is achieved. There we have

$$\begin{array}{l}\lambda x(1) \ldots x(r) u\ v\ z(1) \ldots z(m) \lambda w \lambda w(1) \ldots w(t+3)\\ D(1)(\ldots (D(d)\\ \qquad (x\ X(1) \ldots X(t) u\ v\ z(1) \ldots z(m)\ w\ w(1) \ldots w(t+3))) \ldots) \text{ in M}\end{array}$$

$$\begin{array}{l}\lambda y(1) \ldots y(s) u\ v\ z(1) \ldots z(n) \lambda w \lambda w(1) \ldots w(t+3)\\ D(1)(\ldots (D(d)\\ \qquad (y\ Y(1) \ldots Y(t) u\ v\ z(1) \ldots z(n)\ w\ w(1) \ldots w(t+3))) \ldots) \text{ in N}\end{array}$$

Let $g = s - r$. We apply the corresponding terms to

$$\begin{array}{c}A \ldots A\ B\ A \ldots A(\lambda o \lambda p \lambda x(1)\,\hat{}\,o \ldots \lambda x(t)\,\hat{}\,0\ \lambda u\,\hat{}\,p\ \lambda v\,\hat{}\,p.\ v)\ K \ldots\\ (T(\bigwedge p\ p \to (p \to p))K)\ (T(\bigwedge p\ p \to (p \to p))K^*)\\ (\lambda x(1)\,\hat{}\,(\bigwedge p\ p \to (p \to p)) \ldots \lambda x(t+2)\,\hat{}\,(\bigwedge p\ p \to (p \to p)).\ x(t+2))\ ,\end{array}$$

where there are 1-1 copies of A, B occurs in the position $s + 2$, and there are $t = 1$ occurrences of K, and T is as in Case (i).

In M this beta reduces to

$$A\ X'(1)\ldots X'(t)A\ A\ldots B\ldots$$

and this reduces to

$$\begin{gathered}A\ X'(1)\ldots X'(t)\ A\ A(\lambda o\lambda p\lambda x(1)\hat{}\,0\ldots\lambda x(t)\hat{}\,0\ \lambda u\hat{}\,p\ \lambda v\hat{}\,p,\ K\ldots\\ (T(\bigwedge p\ p\to(p\to p))K)\ (T(\bigwedge p\ p\to(p\to p)K^*)\\ (\lambda x(1)\hat{}\,(\bigwedge p\ p\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2)))\end{gathered}$$

and

$$\begin{gathered}A\ K\ldots(T(\bigwedge p\ p\to(p\to p))K)\ (T(\bigwedge p\ p\to(p\to p))K^*)\\ (\lambda x(1)\hat{}\,(\bigwedge p\ p\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2))\end{gathered}$$

and

$$K^*\ .$$

In N this beta reduces to

$$A\ Y'(1)\ldots Y'(t)A\ B\ldots\ldots$$

and this reduces to

$$\begin{gathered}A\ Y'(1)\ldots Y'(t)\ A\ B(\lambda o\lambda p\lambda x(1)\hat{}\,o\ldots\lambda x(t)\hat{}\,0\lambda u\hat{}\,p.\ v)\ K\ldots\\ (T(\bigwedge p\ p\to(p\to p))K)\ (T(\bigwedge p\ p\to(p\to p))K^*)\\ (\lambda x(1)\hat{}\,(\bigwedge p\ p\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2))\end{gathered}$$

and

$$\begin{gathered}B\ K\ldots(T(\bigwedge p\ p\to(p\to p))K)\ (T(\bigwedge p\ p\to(p\to p))K^*)\\ (\lambda x(1)\hat{}\,(\bigwedge p\ p)\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2))\end{gathered}$$

and

$$\begin{gathered}(\lambda x(1)\hat{}\,(\bigwedge p\ p\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2))\\ K\ldots(T(\bigwedge p\ p\to(p\to p))K)\ (T\bigwedge p\ p\to(p\to p))K^*)\end{gathered}$$

and

$$K\ .$$

Case (iii)

Case (iii) but neither (i) nor (ii) apply. In this case we must descend to depth f in a slightly different way. At # we have

$$\begin{aligned}&\lambda x(1)\ldots x(r)u\ v\ z(1)\ldots z(m)\lambda w\lambda w(1)\ldots w(t+3)\\&D(1)(\ldots(D(d)\\&\qquad(x\ X(1)\ldots X(t)u\ v\ z(1)\ldots z(m)\ w\ w(1)\ldots w(t+3)))\ldots)\text{ in M}\end{aligned}$$

$$\begin{aligned}&\lambda y(1)\ldots y(r)u\ v\ z(1)\ldots z(n)\lambda w\lambda w(1)\ldots w(t+3)\\&D(1)(\ldots(D(d)\\&\qquad(y\ Y(1)\ldots Y(t)u\ v\ z(1)\ldots z(n)\ w\ w(1)\ldots w(t+3)))\ldots)\text{ in N}\end{aligned}$$

and x and y are bound by lambdas in labels at dissimilar nodes in the term trees, or at similar nodes but at a different position in the prefix part of the label, and one or both of these nodes can be #. In the descent, if the lambda corresponding to x is encountered we use A as above but for y we use B. Note that this has no effect above # since both u and v have become A. This procedure should also be followed for the first r lambdas at #. At this point, we have

$$\begin{aligned}&\lambda u\ v\ z(1)\ldots z(m)\lambda w\lambda w(1)\ldots w(t+3)\\&\quad D(1)(\ldots(D(d)\\&\qquad(A\ X'(1)\ldots X'(t)u\ v\ z(1)\ldots z(m)\ w\ w(1)\ldots w(t+3)))\ldots)\text{ in M}\end{aligned}$$

$$\begin{aligned}&\lambda u\ v\ z(1)\ldots z(n)\lambda w\lambda w(1)\ldots w(t+3)\\&\quad D(1)(\ldots(D(d)\\&\qquad(B\ Y'(1)\ldots Y'(t)u\ v\ z(1)\ldots z(n)\ w\ w(1)\ldots w(t+3)))\ldots)\ldots)\text{ in N}\end{aligned}$$

and we can proceed as in Case (ii) by applying both to

$$\begin{aligned}&A\ B(\lambda o\lambda p\lambda x(1)\hat{}\,o\ldots x(t)\hat{}\,0\lambda u\hat{}\,p\lambda v\hat{}\,p.\ v)\ K\ldots\\&\qquad(T(\textstyle\bigwedge p\ p\to(p\to p))K)\ (T(\bigwedge p\ p\to(p\to p))K^*)\\&\quad(\lambda x(1)\hat{}\,(\textstyle\bigwedge p\ p\to(p\to p))\ldots\lambda x(t+2)\hat{}\,(\bigwedge p\ p\to(p\to p)).\ x(t+2))\ ,\end{aligned}$$

where T is as in Case (i). This completes Case (iii).

Thus, in every case if $M = N$ then $K = K^*$. Hence $M = N$ can have no model with a type having more than 1 element. This completes the proof.

References

1. Barendegt, H.P.: The Lambda Calculus. North-Holland, Amsterdam (1981)
2. Böhm, C.: Alcune proprieta delle forme beta-eta-normali nei lambda-K-calcolo. P.I.A.C. 696 (1968)

3. Church, A.: Calculi of Lambda Conversion. P.U.P (1941)
4. de Vrijer, R.: Extending the lambda calculus with surjective pairing is conservative. LICS 4, 204–215 (1989)
5. Girard, J.Y.: Interpretation fonctionelle et elimination des coupures dans l'arithmetique d'ordre supieure. U. Paris VII (1972)
6. Jacopini, G.: A condition for identifying two elements of whatever model of combinatory logic. L.N.C.S. 37 (1975)
7. Reynolds, J.: Personal communication (circa 1985)
8. Statman, R.: Surjective pairing revisited. In: van Oostrom, K., van Raamsdonk (eds.) Liber Amicorum for Roel de Vrijer, University of Amsterdam, Amsterdam (2009)
9. Stovering, K.: Extending the extensional lambda calculus with surjective pairing is conservative. LMCS 2, 1–14 (2006)
10. Mitchell, J.: Polymorphic type inference and containment. Information and Computation 76(2-3), 211–249 (1988)

A Appendix

Here we show that the theorem fails for polymorphic types with products instead of polymorphic types closed under pairing. For this we add the type constructor & and the following clauses to the definitions of types and terms.

- If a and b are types then so is a & b.
- If X has type a and Y has type b then $\langle X, Y\rangle$ has type a & b.
- If X has type a & B then LX has type a and RX has type b.

Proposition 1. *There is no typing of $\lambda x.\ Lx$ and $\lambda x.x$ for which these terms have the same type and are inconsistent.*

Proof. First we suppose that these terms have been typed with the same type a. We shall adopt some notation for sequences. We write p^* for $p(l)\ldots p(t)$ with $lh(p^*) = t$. We adopt also Mitchell's containment relation $a < b \Leftrightarrow a = \bigwedge p^*\ c$ and $b = \bigwedge q^*\ [d^*/p^*]c$ where $lh(d^*) = lh(p^*)$. Then there exist d and e s.t.

- $a = \bigwedge p^*\ (b \to c)$;
- $b < d\ \&\ e$;
- $c = \bigwedge q^*\ d$;
- $b < \bigwedge q^*\ D$.

Consider the leftmost path in b. There cannot be a → along that path since the shallowest such → must be the same depth in both b and c. Thus, this path consists of a sequence of $\bigwedge$'s and &'s ending in a variable, say r. Now r must be bound by a $\bigwedge$ in the prefix of b for otherwise it is at the same depth in both b and c. Thus, there exists an integer m such that $\lambda x.\ L\hat{}m\ x$ has type $b \to \bigwedge p\, p$. Now consider a normal term X of type b, using only the free variables $x\hat{}p$ and $y\hat{}p$ and the constant C of type $\bigwedge p\, p$. The leftmost path in X consists of a sequence of $\langle , \rangle$'s, possibly ending in a single type abstraction, followed by a sequence of L's, R's, and applications. It can only end in C. Thus, we can apply Jacopini's theorem as above to show that $\lambda x.\ Lx = \lambda x.\ x$ has a model with no empty types. End of Proof.

Querying the Fragments of English

Camilo Thorne

KRDB Research Centre for Knowledge and Data,
3 Piazza Domenicani, 39100, Bolzano, Italy
cthorne@inf.unibz.it

Abstract. Controlled languages are fragments of natural languages stripped clean of lexical, structural and semantic ambiguity. They have been proposed as a means for providing natural language front-ends to access structured knowledge sources, given that they compositionally and deterministically translate into the (logic-based) formalisms such back-end systems support. An important issue that arises in this context is the semantic data complexity of accessing such information (i.e., the computational complexity of querying measured w.r.t. the number of instances declared in the back-end knowledge base or database). In this paper we study the semantic data complexity of a distinguished family of context-free controlled fragments, viz., Pratt and Third's fragments of English. In doing so, we pinpoint those fragments for which the reasoning problems are tractable (in PTIME) or intractable (NP-hard or CONP-hard).

Keywords: Controlled language interfaces, computational semantics, semantic data complexity, resolution proof procedures, knowledge base query answering and satisfiability.

1 Introduction

Natural language can be considered the ultimate knowledge representation language, due to its expressiveness and the little prior training required. It is also, arguably, the language casual users prefer when interacting with structured data management systems, i.e., when declaring a data constraint, adding a piece of data or querying such information [11]. These considerations have motivated, from the 1970's and 80's [18] to this day [5,13] a big body of research on natural language interfaces to structured data management systems, such as knowledge-based systems, ontology-based systems and relational databases. But any system that aims at understanding natural language has to deal with the computational costs of semantic processing, viz., with the *semantic complexity* of natural language [16].

Traditionally, the main challenge that semantic processing faces is *ambiguity*, viz., the fact that the same natural language surface form may stand for different deep forms or semantic representations. Ambiguity arises at different levels: at a lexical level (the same word may have different meanings), a structural level (the same sentence may admit difference parses) and a semantic level (the same sentence may "underspecify" several semantic interpretations). Ambiguity induces a very high semantic complexity: in general, the number of readings of an utterance will grow exponentially in its size [10]. Worse still, since natural language semantics is captured, modulo so-called

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 257–271, 2011.

compositional translations $\tau(\cdot)$, by higher order logic (HO) and its fragments, notably, first oder logic (FO), the problem of ambiguity is, in general, undecidable.

To tackle the high semantic complexity of ambiguity it has been proposed to trade off expressiveness, by considering *controlled languages*, viz., ambiguity-free fragments of natural languages which allow for efficient (polynomial) deterministic parsing and semantic interpretation [7], thus guaranteeing high or absolute precision and recall at the cost of constraining natural language input.

Ambiguity unfortunately, is not the only factor inducing prohibitive semantic complexity. In general, we are also interested in *reasoning* over semantic representations. In knowledge-based systems, for instance, we might want to leverage on intensional information (domain knowledge and constraints) to refine queries or questions. Such refinement can be arguably considered part of an utterance's semantic processing. Thus, controlled language coverage (i.e., its verbs, nouns, adjectives, relative clauses, coordinating particles, quantification, etc.) can also impact on semantic complexity. For instance, function words which map to logical operations such as Boolean conjunction and negation can give rise to "Boolean-closed" fragments for which reasoning is intractable [17]. Similar results hold for fragments covering some distinguished classes of (generalized) quantifiers, which are intractable w.r.t. to FO finite model checking [21].

In this paper we study the computational *data complexity* of answering queries over controlled English knowledge bases, viz., the computational complexity of this problem (a FO entailment problem) measured in the number of facts stored in the knowledge base. Data complexity is an important (theoretical) measure of query evaluation efficiency in knowledge-based systems [2]. We consider for this purpose the so-called *fragments of English*, a family of very simple controlled fragments proposed by Pratt and Third in [17]. We show for which fragments data complexity is tractable, for which it is intractable and for which query answering is undecidable. As the reader shall see, bad computational properties arise even in the presence of fragments of very little expressiveness.

2 The Fragments of English and Semantic Complexity

In this section we present a generic methodology for defining controlled fragments, proposed by Pratt and Third in [17], the *fragments of English*. These fragments are defined using context-free semantically enriched grammars and has the advantage of generating, alongside the set of grammatical utterances of the fragment, their logical (HO and FO) meaning representations.

Their semantics follows the Montague semantics paradigm outlined in [14], in which English utterances S and sets $\mathcal{S}$ thereof are mapped into into a HO or FO *meaning representation* $\tau(S)$ or set $\tau(\mathcal{S})$ of formulas, leveraging on the λ-abstraction, application and normalization of type-theory. Semantic actions recursively define the compositional translation $\tau(\cdot)$ on a fragment's sentence constituents.

As with all controlled languages, ambiguity is ruled out: complete utterances admit one and only one parse tree and one and only one semantic representation. Consequently, each fragment of English expresses a unique FO fragment, whose computational properties can be easily studied. In addition, the definition of $\tau(\cdot)$ over grammar lexicons gives rise to their partition into:

- An arbitrarily large *content lexicon* denoting *individuals* with proper nouns ($\mathbf{Pn}$s like "Max"), and unary, binary and ternary *relations*, with, resp., nouns ($\mathbf{N}$s like "beer"), transitive verbs ($\mathbf{TV}$s like "drinks") and ditransitive verbs ($\mathbf{DTV}$s like "gives").
- A finite *function lexicon* denoting *logical operations* over individuals and relations. Relative pronouns ($\mathbf{Relp}$s like "who") and coordinators ($\mathbf{Crd}$s like "and") denote logical conjunction. Determiners ($\mathbf{Det}$s like "some") denote quantification. Negation ($\mathbf{Ng}$s like "not") denotes logical negation. Anaphoric pronouns express co-references between logical expressions.

The fragments themselves are defined incrementally, starting from a base fragment, COP (for "copula"), whose coverage is subsequently expanded to $\mathbf{TV}s$, $\mathbf{DTV}$s, $\mathbf{Relp}$s and (restricted) anaphora, as summarized by Table 1. In what follows we will describe in detail only COP. The other fragments are defined similarly:

Function Lexicon	
$\mathbf{Det} \longrightarrow$ every	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\forall x^{e}(P(x) \Rightarrow Q(x))$
$\mathbf{Det} \longrightarrow$ some	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\exists x^{e}(P(x) \wedge Q(x))$
$\mathbf{Det} \longrightarrow$ no	$\tau(\mathbf{Det}) := \lambda P^{e\to t}.\lambda Q^{e\to t}.\forall x^{e}(P(x) \Rightarrow \neg Q(x))$
$\mathbf{Ng} \longrightarrow$ not	$\tau(\mathbf{Ng}) := \lambda P^{e\to t}.\lambda x^{e}.\neg P(x)$

Phrase Structure Rules		Content Lexicon	
$\mathbf{S}\to\mathbf{NP}\ \mathbf{VP}$	$\tau(\mathbf{S}):=\tau(\mathbf{NP})(\tau(\mathbf{VP}))$	$\mathbf{N}\to$woman	$\tau(\mathbf{N}):=\lambda x^{e}.\mathit{Woman}(x)$
$\mathbf{VP}\to$ is a $\mathbf{N}$	$\tau(\mathbf{VP}):=\tau(\mathbf{N})$	$\mathbf{N}\to$man	$\tau(\mathbf{N}):=\lambda x^{e}.\mathit{Man}(x)$
$\mathbf{VP}\to$ is $\mathbf{Ng}$ a $\mathbf{N}$	$\tau(\mathbf{VP}):=\tau(\mathbf{Ng})(\tau(\mathbf{N}))$	$\mathbf{N}\to$person	$\tau(\mathbf{N}):=\lambda x^{e}.\mathit{Person}(x)$
$\mathbf{NP}\to\mathbf{Pn}$	$\tau(\mathbf{NP}):=\tau(\mathbf{Pn})$	$\mathbf{N}\to$human	$\tau(\mathbf{N}):=\lambda x^{e}.\mathit{Human}(x)$
$\mathbf{NP}\to\mathbf{Det}\ \mathbf{N}$	$\tau(\mathbf{NP}):=\tau(\mathbf{Det})(\tau(\mathbf{N}))$	$\mathbf{Pn}\to$Mary	$\tau(\mathbf{Pn}):=\lambda P^{e\to t}.P(\text{Mary})$

The function lexicon express FO universal and existential quantification, in addition to a form of negation. The content lexicon may contain an arbitrarily large stock of common and proper nouns specifying an arbitrarily large FO signature of unary predicates and individual constants.

To each lexical entry and each grammar rewriting rule a semantic action is associated and the computation of $\tau(\cdot)$ can be done on the fly, i.e., side-by-side with the computation of the parse tree. See Figure 1. In COP we can express FO sentences of these forms:

$\mathit{Woman}(\text{Mary})$	Mary is a woman.	$\forall x(\mathit{Man}(x) \Rightarrow \mathit{Person}(x))$	Every man is a person.
$\neg\mathit{Man}(\text{Mary})$	Mary is not a man.	$\forall x(\mathit{Woman}(x) \Rightarrow \neg\mathit{Man}(x))$	No woman is a man.
$\forall x(\mathit{Person}(x) \Rightarrow \mathit{Human}(x))$	Every person is a human.		
$\exists x(\mathit{Person}(x) \wedge \mathit{Woman}(x))$	Some person is a woman		
$\exists x(\mathit{Person}(x) \wedge \neg\mathit{Woman}(x))$	Some person is not a woman.		

More in general, by suitably modifying the content lexicon, we can express *any* FO sentence of these forms. This defines a proper fragment of FO, which we may denote by abuse of notation COP as well [17]. Semantic complexity will be, in general, correlated to the fragment's function lexicon.

Table 1. The fragments of English

COP	copula, common and proper nouns, negation universal and existential quantifiers
COP+TV	COP + transitive verbs (“loves”)
COP+TV+DTV	COP + transitive and ditransitive verbs (“gives”)
COP+Rel	COP + relative pronouns (“who”, “that”)
COP+Rel+TV	COP+Rel + transitive verbs
COP+Rel+TV+DTV	COP+Rel + transitive and ditransitive verbs
COP+Rel+TV+RA	COP+Rel+TV + restricted anaphora (“he”, “it”, “herself”)

Modulo compositionality and the ensuing logic meaning representations, the fragments of English can be said to be subsumed by several known fragments of FO: the monadic fragment of FO (COP and COP+Rel), the 2-variable fragment of FO (due to transitive verbs: COP+TV, COP+Rel+TV and COP+Rel+TV+RA) and the 3-variable fragment of FO (due to ditransitive verbs: COP+TV+DTV and COP+Rel+TV+DTV). They are known to be subsumed or to overlap with several knowledge representation logics such as description logics [4] and have been shown to be decidable for satiafiability [17]. It is perhaps interesting to note that the non-monadic fragments are incomparable to other known fragments such as the guarded fragment[1].

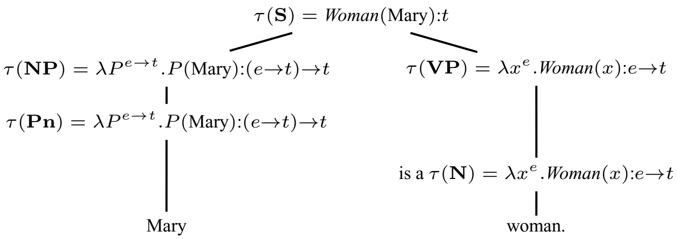

$\tau(\mathbf{S}) = \mathit{Woman}(\text{Mary}){:}t$
$\tau(\mathbf{NP}) = \lambda P^{e\to t}.P(\text{Mary}){:}(e\to t)\to t$
$\tau(\mathbf{VP}) = \lambda x^{e}.\mathit{Woman}(x){:}e\to t$
$\tau(\mathbf{Pn}) = \lambda P^{e\to t}.P(\text{Mary}){:}(e\to t)\to t$
is a $\tau(\mathbf{N}) = \lambda x^{e}.\mathit{Woman}(x){:}e\to t$
Mary
woman.

Fig. 1. Parse tree for the COP sentence “Mary is a woman”

In general, controlled fragments designed with practical applications in mind tend to be very expressive and are not easily amenable to a computational complexity analysis. The best-known controlled fragment of English, Attempto Controlled English, ACE [7] (that contains all of the fragments discussed here), is for instance more expressive than FO itself, thus being undecidable for satisfiability. Pratt and Third’s fragments by contrast, by being simple and decidable (in most cases), allow to pinpoint those combinations of English constructs that give rise to tractable and intractable computational complexity.

3 Querying the Fragments of English

In the remainder of this paper, we will focus on one particular kind of data access and storage system, an ontology-based system or knowledge base, in which data, stored as

[1] A COP+TV sentence like “Every liar knows every trick.”, yields a semantic representation, $\forall x(\mathit{Liar}(x) \Rightarrow \forall y(\mathit{Trick}(y) \Rightarrow \mathit{Knows}(x,y)))$, that is not guarded [8].

instances or as records in a (relational) database is managed and queried through an intensional middle layer of *domain constraints* or *ontology* Σ (a set of FO sentences), expressed by a set $\mathcal{S}$ of controlled English sentences.

A *knowledge base* is a pair (Σ, Δ), with Σ an ontology and Δ a a set of facts or FO relational ground atoms. The integer $\#(\Delta)$ denotes the *size* of Δ, the number facts it contains. The usual FO notions of truth and satisfaction apply to ontologies, databases and knowledge bases, and to their constraints and facts.

A *conjunctive query* is an existentially quantified conjunction of positive FO relational atoms

$$\varphi(\boldsymbol{x}) := \exists \boldsymbol{y} \varphi(\boldsymbol{x}, \boldsymbol{y})$$

over variables $\boldsymbol{x}$ and $\boldsymbol{y}$, where the free variables $\boldsymbol{x}$ are known also as the CQ's *distinguished variables*. A *union of conjunctive queries* is a finite disjunction

$$\varphi(\boldsymbol{x}) := \exists \boldsymbol{y}_1 \varphi_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \cdots \vee \exists \boldsymbol{y}_k \varphi_k(\boldsymbol{x}, \boldsymbol{y}_k)$$

of conjunctive queries. A *tree-shaped query* (TCQ) is a CQ with one distinguished variable x, called *root*, defined inductively by

$$\varphi(x) \longrightarrow A(x) \mid \exists y R(x, y) \mid \exists y R(x, y) \wedge \varphi(y) \mid \varphi(x) \wedge \varphi'(x)$$

The *length* $|\boldsymbol{x}|$ of the sequence of distinguished variables is known as the *arity* of the U(T)CQ. U(T)CQs are said to be *Boolean* when they contain no free variables.

Note that U(T)CQs are positive existential FO formulas. U(T)CQs have a practical motivation: they correspond to the SELECT-PROJECT-JOIN fragment of SQL, the most frequently used class of queries in relational databases [1].

A *substitution* σ is a function from variables to terms. It is called a *renaming* when it is a function from variables to variables and a *grounding* when it is a function from variables to constants. Substitutions can be extended to complex syntactic objects in the standard way.

A knowledge base (Σ, Δ) is said to entail a U(T)CQ φ w.r.t. a substitution σ, in symbols, $\Sigma \cup \Delta \models \varphi\sigma$, if every model of $\Sigma \cup \Delta$ is a model of $\varphi\sigma$ (i.e., plain FO entailment).

Two reasoning tasks are of relevance: *(i)* Consistency: we want to know whether the data being stored is not in conflict with the constraints and is meaningful. *(ii)* Query answering: we want to retrieve information. In both cases, it is customary to understand the data complexity of the tasks, viz., the computational complexity of their associated recognition decision problems measured w.r.t. the number of instances or records stored in the system. This idea was first proposed by Vardi in [23] for relational databases, and is based on the observation that the main issue in real-world applications is scalability to massive datasets.

Given a knowledge base (Σ, Δ), the knowledge base *satisfiability* problem (KBQA) is the decision problem defined by: **Input:** the set Δ of facts. **Question:** does there exist a model for $\Sigma \cup \Delta$?

Given a knowledge base (Σ, Δ), a U(T)CQ φ of distinguished variables $\boldsymbol{x}$, a sequence of constants $\boldsymbol{c}$ and a substitution σ s.t., $\sigma(\boldsymbol{x}) = \boldsymbol{c}$, the knowledge base *query answering* problem (KBQA) is the decision problem defined by: **Input:** the set Δ of facts. **Question:** does $\Sigma \cup \Delta \models \varphi\sigma$ hold?

4 Resolution Saturations and Data Complexity

The resolution calculus was first proposed by Robinson in [19] as a sound and complete, but not necessarily terminating machine-oriented calculus for FO. Later, Joyner in [9] showed that resolution can be turned into a decision procedure for (un)satisfiability for a number of distinguished fragments of FO, by proposing several saturation- or forward-chaining-based *refinings* of resolution. In [22], we showed how such refinements can be used to provide upper data complexity bounds for KBSAT for several fragments of English. In this section, we extend such results to all the (decidable) fragments of English and generalize the technique to KBQA.

A *term* t is *(i)* a variable x or a constant c or *(ii)* an expression $f(t_1,\ldots,t_n)$ where f is a function symbol and $t_1,\ldots,t_n$ are terms. In the latter case, we speak about *function terms*. A *literal* L is a FO atom $P(t_1,\ldots,t_n)$. By a *clause* we understand a disjunction $L_1 \vee \cdots \vee L_n \vee \overline{N}_{n+1} \vee \cdots \vee \overline{N}_{n+m}$ of *positive* and *negative* literals. The *empty* clause or *falsum* is denoted $\perp$. By $\mathit{Var}(t)$, $\mathit{Var}(L)$ and $\mathit{Var}(C)$ we denote the sets of variables of, resp., term t, literal L and clause C. A term, literal, clause or set of clauses is said to be *ground* if it contains no variables.

A *unifier* of two terms t and t' is a substitution σ s.t. $t\sigma = t'\sigma$. A *most general unifier* is a unifier σ s.t. for every other unifier σ' there exists a renaming σ'' with $\sigma' = \sigma\sigma''$.

The *depth* of a term is defined recursively by *(i)* $d(x) := d(c) := 0$ and *(ii)* $d(f(t_1,\ldots,t_n)) := \max\{d(t_i) \mid i \in [1,n]\} + 1$. The *depth* $d(L)$ of a literal L or $d(\Gamma)$ of a set of clauses Γ is the maximal depth of their terms. The *relative depth* of a variable x in a term t is defined by *(i)* $d(x,t) = 0$ if $x \notin \mathit{Var}(t)$, otherwise *(ii)* $d(x,x) := 1$ and $d(x, f(t_1,\ldots,t_n)) := \max\{d(x,t_i) \mid i \in [1,n]\} + 1$. The *relative depth* $d(x,L)$ of a variable x in a literal L is its maximal relative depth among L's terms.

A clause C is "well behaved" whenever *(i)* $\mathit{Var}(L) \leq 1$ and *(ii)* either for every functional term t in C, $\mathit{Var}(t) = \mathit{Var}(C)$, or $\mathit{Var}(L) = \mathit{Var}(C)$, for all literals in C[2].

Table 2. Resolution calculi. Calculus $\mathcal{R}_{2,5}$ is a decision procedure for "well-behaved clauses [6].

		split	*mon*	*split, mon*
	$\mathcal{R}_{1,1}$	$\mathcal{R}_{1,2}$	$\mathcal{R}_{1,4}$	$\mathcal{R}_{1,5}$
$\prec_d$	$\mathcal{R}_{2,1}$	$\mathcal{R}_{2,2}$	$\mathcal{R}_{2,4}$	$\mathcal{R}_{2,5}$

The saturation-based versions of the (ordered) resolution calculus iteratively (monotonically w.r.t. $\subseteq$) generate the set of all possible clauses derived from Γ, until either *(i)* $\perp$ is derived or *(ii)* all possible clauses are generated (fixpoint computation). Such clauses are obtained using *res* (resolution), *fact* (factoring) and *split* (splitting):

[2] Clauses that are "well-behaved" correspond to the clauses from the S^+ class studied in [6].

$$\textit{res}\ \frac{C \lor \overline{L} \qquad C \lor L'}{(C \lor C')\sigma} \qquad\qquad \textit{fact}\ \frac{C \lor L \lor L'}{(C \lor L)\sigma}$$

$$\textit{split}\ \frac{C \lor L \lor L' \qquad \begin{array}{c} C \lor L \\ \vdots \\ C'\sigma \end{array} \qquad \begin{array}{c} C \lor L' \\ \vdots \\ C'\sigma \end{array}}{C'\sigma}\ (\mathit{Var}(L) \cap \mathit{Var}(L') = \emptyset)$$

where σ is a most general unifier of L and L', the *acceptable* depth ordering (well-founded and substitution-invariant partial order on clause literals and sets thereof) $\prec_d$ defined by

$$L \prec_d L' \quad \text{iff} \quad \textit{(i)}\ d(L) < d(L'),\ \textit{(ii)}\ \mathit{Var}(L) \subseteq \mathit{Var}(L'),\ \textit{(iii)}\ d(x, L) < d(x, L'),$$

for all $x \in \mathit{Var}(L)$, and the *mon* (monadization) rule. Orderings give rise to the ordered versions of *res*, *fact* and *split*. Note that *split* introduces branching in saturations.

The $\prec_d$ ordering prevents clause depth from "well-behaved" clauses from growing beyond a finite bound $d \geq 0$, whereas splitting prevents their length l from growing arbitrarily. Now, ordered *res*, on "well-behaved" clauses may generate non-"well-behaved" clauses, whose depth may grow arbitrarily. Given a set Γ of clauses derived by ordered resolution from "well-behaved" clauses, *mon* substitutes, in each literal L in Γ, all the variables $x \in \mathit{Var}(L)$ that *do not* occur in L's functional terms t, with either t or the constants c that occur in Γ. Thus, monadization, which is satisfiability-preserving for such clauses (but not necessarily in the general case), ensures that "well-behaved" clauses are closed under the ordered rules, guaranteeing the termination of saturations [9,6].

Formally, consider a derivation function $\rho(\cdot)$ over sets of clauses, defined in terms of rules stated above. A *resolution calculus* is a function $\mathcal{R}(\cdot)$ s.t. $\mathcal{R}(\Gamma) := \Gamma \cup \rho(\Gamma)$. A *saturation* is defined as the limit Γ^∞ of the sequence defined by *(i)* $\mathcal{R}^0(\Gamma) := \Gamma$ and *(ii)* $\mathcal{R}^{i+1}(\Gamma) := \mathcal{R}(\mathcal{R}^i(\Gamma))$, for $i > 0$. The positive integer i is called the *depth* of the saturation. Different calculi arise from the different combinations of rules, orderings and refinements, as summarized by Table 2.

Resolution is sound and complete w.r.t. (un)satisfiability, in the sense that Γ is unsatisfiable iff $\bot \in \Gamma^\infty$, and $\mathcal{R}_{2,5}$ in particular terminates if Γ is a set of "well behaved" clauses. Moreover, it is immediate to see that, in general, resolving upon ground clauses can be delayed to the last steps of the saturation:

Proposition 1. *Given a terminating saturation for a set Γ of clauses containing ground atoms, one can delay deriving upon those ground atoms to the last steps of the saturation.*

Saturations exhibit the shape of a tree (of branching factor 2) or of a sequence, depending on whether the calculi make use or not of the splitting rule. See Figure 2. Due to this observation, resolution decision procedures can be used to provide upper data complexity bounds.

Lemma 1 (from [22], Lem. 1). *Let Γ be a set of clauses containing n constants (and finitely many function and relation symbols) for which there exist both a term depth*

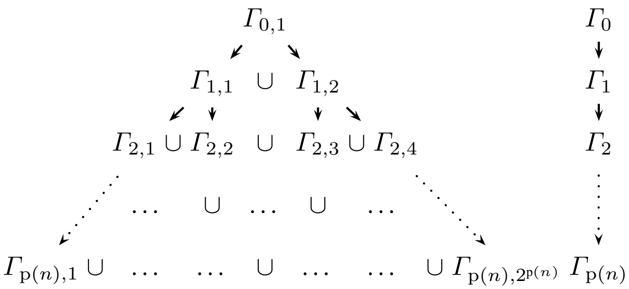

$\Gamma_{0,1}$
$\Gamma_{1,1} \cup \Gamma_{1,2}$
$\Gamma_{2,1} \cup \Gamma_{2,2} \cup \Gamma_{2,3} \cup \Gamma_{2,4}$
$\ldots \cup \ldots \cup \ldots$
$\Gamma_{\mathrm{p}(n),1} \cup \ldots \ldots \cup \ldots \ldots \cup \Gamma_{\mathrm{p}(n),2^{\mathrm{p}(n)}}$
Γ_0
Γ_1
Γ_2
$\Gamma_{\mathrm{p}(n)}$

Fig. 2. Terminating saturations of a set Γ of clauses using (tree-shaped) and not using (linear-shaped) the splitting rule. In the figure, directed edges denote derivation steps and $\mathrm{p}(\cdot)$ denotes a polynomial on the number n of constants in Γ, that bounds the depth of saturation.

bound $d \geq 0$ and a clause length bound $k \geq 0$. In the worst case: (i) the number of clauses derivable by the saturation is (a) exponential in n if we use the splitting rule or (b) polynomial in n otherwise, and (ii) the depth of the saturation is polynomial in n.

4.1 Tractable Fragments

In this section we show that the data tractable fragments for KBQA are those involving no relative pronouns and that are not Boolean-closed. Similar results hold for KBSAT, provided relatives and transitive verbs are not covered simulaneously by the same fragment(s). For the PTIME upper bound below we use saturations to define a reduction to the well-known (un)satisfiability problem for the HORN fragment (the fragment of FO clauses in which at most one literal is positive), which is known to be in PTIME in the ground case (see, e.g., [8]).

Theorem 1. *The data complexity of* KBSAT *is in* PTIME *for COP+Rel.*

Proof. Let Σ be a set of COP+Rel meaning representations and Δ a database, where Σ is fixed and let $\Sigma^{cl} \cup \Delta^{cl}$ be their clausification and Skolemization. $\Sigma^{cl} \cup \Delta^{cl}$ can be computed in $\mathbf{O}(\log \#(\Delta))$ space. This gives way to polynomially many constants in $\#(\Delta)$. Now, the clauses in $\Sigma^{cl} \cup \Delta^{cl}$ are *(i)* monadic, and *(ii)* if in Σ^{cl}, Boolean combinations of unary atoms over the single variable x, and containing no function symbols. Clearly, the clauses in $\Sigma^{cl} \cup \Delta^{cl}$ are "well-behaved" clauses of the kind for which resolution saturations can be used as a decision procedure. Furthermore, the *split* rule is not applicable. Use therefore the calculus $\mathcal{R}_{2,4}$: by Lemma 1, we know that such a procedure will run in time polynomial in the number constants of $\Sigma^{cl} \cup \Delta^{cl}$ and hence in time polynomial in $\#(\Delta)$. □

Theorem 2 (from [22], Th. 1). *The data complexity for* KBSAT *is in* LSPACE *for COP, COP+TV and COP+TV+DTV.*

Theorem 3 (from [22], Th. 2). *The data complexity of* KBQA *for COP and UCQs is in* LSPACE.

Theorem 4. *The data complexity of* KBQA *for the fragments COP+TV+DTV and UCQs is in* PTIME.

Proof. By reduction to HORN satisfiability. Let Σ be a set of fixed COP+TV+DTV meaning representations, Δ a database and φ a fixed UCQ of distinguished variables $\boldsymbol{x}$. Let $\boldsymbol{c}$ be a tuple (a sequence of $|\boldsymbol{x}|$ domain constants) and σ a substitution mapping $\boldsymbol{x}$ to $\boldsymbol{c}$. To know whether $\boldsymbol{c}$ is an answer to φ over (Σ, Δ), it is sufficient to check if $\Sigma \cup \Delta \models \varphi\sigma$.

Consider now the sets Σ^{cl}, Δ^{cl} and $(\neg\varphi\sigma)^{cl}$, where by $\cdot^{cl}$ we denote the (satisfiability-preserving) Skolemization and clausification of Σ, Δ and $\neg\varphi\sigma$. Skolemization and clausification are constant in $\#(\Delta)$: Δ is already a set of (ground) clauses. We claim that Σ^{cl}, Δ^{cl} and $(\neg\varphi\sigma)^{cl}$ are sets of HORN clauses. Clearly, Δ^{cl} is a set of HORN ground clauses. On the other hand, the clauses in Σ^{cl} are all of the form(s) [17]:

$$\neg P(x)\vee\pm Q(x),\ \neg P(x)\vee\pm L(x),\ \neg P(x)\vee Q(f(x)),\ \neg P(x)\vee\neg Q(y)\vee\pm L(x,y),$$
$$\pm L,\ \neg P(x)\vee\neg Q(y)\vee\neg N(z)\pm L(x,y,z),\ \neg P(x)\vee\neg Q(y)\vee N(g(x,y)),$$

where $L(\boldsymbol{x})$ stands for a binary or ternary literal with free variables $\boldsymbol{x}$ and possibly functional terms, which are also HORN. Since φ is a UCQ, this means that $\varphi\sigma$ is of the form $\exists y_1\psi(\boldsymbol{c}, \boldsymbol{y}_1) \vee \cdots \vee \exists y_k\psi(\boldsymbol{c}, \boldsymbol{y}_k)$, with each $\psi_i(\boldsymbol{c}, \boldsymbol{y}_i)$ of the form $L'_{i1}(\boldsymbol{t}_{i1}) \wedge \cdots \wedge L'_{im}(\boldsymbol{t}_{im})$, where $\boldsymbol{c} \cup \boldsymbol{y}_i \subseteq \boldsymbol{t}_{i1} \cup \cdots \cup \boldsymbol{t}_{im}$ and L'_{ij}, for $1 \leq j \leq m$, is a relational symbol of arity ≤ 2. Hence, $(\neg\varphi\sigma)^{cl}$ is the set of (two variable) HORN clauses $\{\neg L'_{11}(\boldsymbol{t}_{11}) \vee \cdots \vee \neg L'_{1m}(\boldsymbol{t}_{1m})\} \cup \cdots \cup \{\neg L'_{k1}(\boldsymbol{t}_{k1}) \vee \cdots \vee \neg L'_{km}(\boldsymbol{t}_{km})\}$.

The clauses in Σ^{cl}, Δ^{cl} and $(\neg\varphi\sigma)^{cl}$ are "well-behaved" clauses, viz., clauses where either all literals are essentially monadic, or covering, or whose functional terms are covering. Hence, resolution saturations (e.g., calculus $\mathcal{R}_{2,5}$) can be used to decide their (un)satisfiability. Furthermore, by Proposition 1, we know that we can "separate" facts from non-ground clauses. Finally, by grounding the saturation (following the Herbrand theorem, cf. [12], Prop. IV-5), we can reduce answering a question to checking for the satisfiability of a set of HORN (propositional) clauses, since

$$\begin{array}{lll} \Sigma \cup \Delta \models \varphi\sigma & \text{iff} & \Sigma^{cl} \cup \Delta^{cl} \cup (\neg\varphi\sigma)^{cl} \text{ is unsatisfiable} \\ & \text{iff} & \bot \in (\Sigma^{cl} \cup \Delta^{cl} \cup (\neg\varphi\sigma)^{cl})^{\infty} \\ & \text{iff} & \bot \in ((\Sigma^{cl} \cup \Delta^{cl})^{\infty} \cup (\neg\varphi\sigma)^{cl})^{\infty} \\ & \text{iff} & \mathrm{GR}((\Sigma^{cl} \cup (\neg\varphi\sigma)^{cl})^{\infty}) \cup \Delta^{cl} \text{ is unsatisfiable,} \end{array} \tag{$\dagger$}$$

where $\mathrm{GR}(\cdot)$ denotes the operation that grounds the clauses in $(\Sigma^{cl} \cup (\neg\varphi\sigma)^{cl})^{\infty})$ (of which there are finitely many) with constants taken from *adom*(Δ), holds.

By (†) we know that the reduction is sound and complete. We already saw that the clausification procedure is constant in $\#(\Delta)$. In addition, by the separation property, saturating, while exponential in general, can be done independently from the data and thus does not affect data complexity and is thus constant in $\#(\Delta)$ We also know that there are polynomially many groundings in $\#(\Delta)$. Finally, checking for the satisfiability of $\mathrm{GR}((\Sigma^{cl} \cup (\neg\varphi\sigma)^{cl})^{\infty}) \cup \Delta^{cl}$, which are HORN, can be done in time polynomial in $\#(\Delta)$. □

Corollary 1. *The data complexity of* KBQA *for the controlled fragments COP+TV, and COP+TV+DTV and UCQs is in* PTIME.

4.2 Intractable Fragments

In this section we show that the data intractable fragments for KBQA are those covering relative pronouns and that are Boolean-closed. This is shown by a reduction from an NP-complete propositional satisfiability problem. For KBSAT, the boundary lies in COP+Rel+TV that covers both relatives and transitive verbs. Membership in, resp., NP and CONP is derived using saturations.

Theorem 5 (from [22], Th. 1). *The data complexity of* KBSAT *is* NP*-complete for COP+Rel+TV and COP+Rel+TV+DTV.*

The satisfiability problem for *propositional 2+2 formulas* (2+2-SAT) is the decision problem defined by: **Input:** a conjunction of clauses of the form form $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ where each clause $\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$ is a disjunction of two positive and two negative literals. **Question:** does there exist a truth assignment $\delta(\cdot)$ s.t. $\delta(\psi) = 1$? This problem was shown to be NP-complete by Schaerf in [20].

Lemma 2. KBQA *is* CONP*-hard in data complexity for COP+Rel and TCQs.*

Proof. We define a reduction from 2+2-SAT. Let $\psi := \psi_1 \wedge \cdots \wedge \psi_k$ be a 2+2-formula over the literals $At(\psi) := \{l_1, ..., l_m\}$ where, for $i \in [1, k]$, $\psi_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$ is a disjunction of two positive and two negative literals.

Let N_1 ("has as first negative literal"), N_2 ("has as second negative literal"), P_1 ("has as first positive literal") and P_2 ("has as second positive literal") four binary relation symbols/transitive verbs. Let A ("literal"), A_f ("false literal") and A_t ("true literal") be three unary relation symbols/nouns. For each clause ψ_i in ψ, for $i \in [1, k]$, introduce an individual constant/proper name c_i and similarly a constant/proper name l for every $l \in At(\psi)$. Encode ψ with the facts

$$\Delta_\psi := \left\{ \begin{array}{c} A(p_{11}), A(p_{12}), A(n_{11}), A(n_{12}), \ldots, A(p_{k1}), A(p_{k2}), A(n_{k1}), A(n_{k2}), \\ P_1(c_1, p_{11}), P_2(c_1, p_{12}), N_1(c_1, n_{11}), P_2(c_1, n_{12}), \ldots, P_1(c_k, p_{k1}), \\ P_2(c_k, p_{k2}), N_1(c_k, n_{k1}), N_2(c_k, n_{k2}) \end{array} \right\},$$

consider, resp., the declarative sentences $\mathcal{S}$

"No A is not an A_t that is not an A_f", "No A_t is an A_f." and "No A_f is an A_t.",

and the TCQ

$$\begin{aligned} \varphi := \exists x \exists y (P_1(x, y) \wedge A_f(y)) \wedge \exists z (P_2(x, z) \wedge A_f(z)) \wedge \\ \exists w (N_1(x, w) \wedge A_t(w)) \wedge \exists v (N_2(x, v) \wedge A_t(v))), \end{aligned}$$

and claim that

$$\psi \text{ is satisfiable} \quad \text{iff} \quad \tau(\mathcal{S}) \cup \Delta_\psi \not\models \varphi \qquad (\dagger)$$

($\Rightarrow$) Suppose that ψ is satisfiable and let $\delta(\cdot)$ be a truth assignment. Then, for all $i \in [1, k]$, $\delta(\psi_i) = 1$, i.e., $\delta(p_{i1}) = 1$ or $\delta(p_{i2}) = 1$ or $\delta(n_{i1}) = 0$ or $\delta(n_{i2}) = 0$. Given this, we can construct an interpretation $\mathcal{I} = (\mathbb{D}_\mathcal{I}, \cdot^\mathcal{I})$ s.t. $\mathcal{I} \models \tau(\mathcal{S}) \cup \Delta_\psi$ but $\mathcal{I} \not\models \varphi$ as follows:

- $\mathbb{D}_{\mathcal{I}} := \{c_i, p_{ij}, n_{ij} \mid i \in [1,k], j = 1,2\}$, $A^{\mathcal{I}} := \{l \in At(\psi) \mid A(l) \in \tau(\mathcal{F}_\psi)\}$,
- $A_f^{\mathcal{I}} := \{l \in A^{\mathcal{I}} \mid \delta(l) = 0\}$, $A_t^{\mathcal{I}} := \{l \in A^{\mathcal{I}} \mid \delta(l) = 1\}$,
- $P_j^{\mathcal{I}} := \{(c_i, p_{ij}) \mid P_j(c_i, p_{ij}) \in \Delta_\psi, i \in [1,k]\}$, and
- $N_j^{\mathcal{I}} := \{(c_i, n_{ij}) \mid N_j(c_i, n_{ij}) \in \Delta_\psi, i \in [1,k]\}$,

($\Leftarrow$) Let $\mathcal{I}$ be a model $\tau(\mathcal{S}) \cup \Delta_\psi$ of s.t. $\mathcal{I}, \gamma \not\models \varphi$ for all γ. We want to show that there exists a truth assignment $\delta(\cdot)$ s.t. $\delta(\psi) = 1$. Let $\delta\colon At(\psi) \to \{0,1\}$ be the truth assignment s.t.

$$\delta(l) = 1 \quad \text{iff} \quad l \in A_t^{\mathcal{I}}.$$

Now, by assumption $\mathcal{I}, \gamma \not\models \varphi$, for all γ. This implies, for all $i \in [1,k]$, that either $p_{i1} \notin A_f^{\mathcal{I}}$ or $p_{i2} \notin A_f^{\mathcal{I}}$ or $n_{i1} \notin A_t^{\mathcal{I}}$ or $n_{i2} \notin A_t^{\mathcal{I}}$. Now, recall that $\mathcal{I} \models \tau(\mathcal{S})$, where $\tau(\mathcal{S})$ contains the axioms $\forall x(A(x) \Rightarrow A_t(x) \vee A_f(x)), \forall x(A_t(x) \Rightarrow \neg A_f(x))$ and $\forall x(A_f(x) \Rightarrow \neg A_t(x))$, that "say" that a literal is either true or false, but not both. Hence if $p_{i1} \notin A_f^{\mathcal{I}}$, then, by definition of $\delta(\cdot)$, $\delta(p_{i1}) = 1$ and similarly for the other cases. Therefore, $\delta(\psi_i) = 1$, for all $i \in [1,k]$, and thus $\delta(\psi) = 1$. □

Lemma 3 (from [22], Th. 2). *If we consider TCQs, the data complexity of* KBQA *is in* coNP *for COP+Rel, COP+Rel+TV and COP+Rel+TV+DTV.*

Theorem 6. *The data complexity of* KBQA *and TCQs is* coNP*-complete for COP+Rel, COP+Rel+TV and COP+Rel+TV+DTV.*

Proof. Follows from Lemma 2 and Lemma 3. □

5 An Undecidable Fragment

As we have seen, adding relatives to COP makes answering U(T)CQs hard w.r.t. data complexity. In this section we will see that when we consider also transitive verbs and restricted anaphoric pronouns (that co-refer with their closest antecedent noun), KBQA becomes undecidable. We will consider a slightly enriched version of the fragment COP+Rel+TV+DTV+RA, COP+Rel+TV+DTV+RA$^+$, that covers verbs in *passive* form (e.g., "is loved by") and *indeterminate pronouns* (e.g., "somebody"). This we prove by a reduction from the unbounded tiling problem.

A *tiling grid* or *grid* is a tuple $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$, where $\mathbb{T} := \{c_0, \ldots, c_k\}$ is a finite set of k *tiles* and $\mathbb{V}$ and $\mathbb{H}$ are binary relations over $\mathbb{T}$, called, resp., the *vertical* and *horizontal* relations. A *tiling* is a function $t\colon \mathbb{N} \times \mathbb{N} \to \mathbb{T}$ that satisfies the *horizontal* and *vertical adjacency constraints*, i.e., s.t., for all $i, j \in \mathbb{N}$, $(t(i,j), t(i,j+1)) \in \mathbb{H}$ and $(t(i,j), t(i+1,j)) \in \mathbb{V}$. The *unbounded tiling problem* (TP) is the decision problem defined by: **Input:** a grid $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$. **Question:** does a tiling t exist for $\mathcal{T}$? It was shown to be undecidable by Berger in [3].

Theorem 7. KBQA *is undecidable for COP+Rel+TV+RA$^+$ and arbitrary CQs.*

Proof. By reduction from TP to the *complement* of KBQA for COP+Rel+TV+RA$^+$ and arbitrary CQs. Let $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$ be a tiling grid, with $\mathbb{T} := \{c_0, \ldots, c_k\}$ being its tiles and $\mathbb{H}$ and $\mathbb{V}$ the, resp., horizontal and vertical relations over $\mathbb{T}$. We can encode $\mathcal{T}$ with a set $\mathcal{S}_{\mathcal{T}}$ of COP+Rel+TV+RA sentences and a set of facts $\Delta_{\mathcal{T}}$ as follows.

Let the transitive verbs/binary relation symbols H and V encode, resp., the horizontal $\mathbb{H}$ and vertical $\mathbb{V}$ relations. Let the noun/unary relation C_i encode a tile $c_i \in \mathbb{T}$, for $i \in [0, k]$. Let finally $\bar{H}$ be a “fresh” transitive verb/binary relation. We start by defining the set $\mathcal{S}_\mathcal{T}$ that encodes the structure of the grid:

Everything Hs something. Everything Vs something. (1)

For all $i \in [0, k]$: Anything that is not a C_i is a C_{i+1}. No C_i is a C_{i+1}. (2)

For all $(c, c') \notin \mathbb{V}$: Everybody who Hs somebody is a C.
Everybody who is Hd by somebody is a C'. (3)

For all $(c, c') \notin \mathbb{H}$: Everybody who Vs somebody is a C.
Everybody who is Vd by somebody is a C'. (4)

Everybody who $\bar{H}$s somebody does not H him.
Everybody who does not H somebody $\bar{H}'$s him. (5)

Next, we define $\Delta_\mathcal{T}$, that encodes the initial state of the grid:

$$\Delta_\mathcal{T} := \left\{ C_0(c_0), \ldots, C_k(c_k), H(c_0, c_1), \ldots, H(c_{k-1}, c_k) \right\}.$$

Finally, we consider the CQ

$$\varphi := \exists x \exists y \exists z \exists w (H(x, y) \wedge V(y, w) \wedge V(x, z) \wedge \bar{H}(z, w)),$$

and claim that

$$\text{there exists a tiling } t \text{ for } \mathcal{T} \quad \text{iff} \quad \tau(\mathcal{S}_\mathcal{T}) \cup \Delta_\mathcal{T} \not\models \varphi. \qquad (\dagger)$$

Modulo $\tau(\cdot)$, the sentences from (1) express $\forall x \exists y H(x, y)$ and $\forall x \exists y V(x, y)$, which “say” that the grid is unbounded. Those from (2), express, resp., $\forall x(C_0(x) \vee \cdots \vee C_k(x))$ and $\forall x \neg (C_0(x) \wedge \cdots \wedge C_k(x))$, viz., that tiles are pairwise distinct. Sentences from (3)–(4), expressing, resp., the FO formulas $\bigwedge \{\forall x(\exists y H(x, y) \Rightarrow C(x)) \wedge \forall x(\exists y H(y, x) \Rightarrow C'(x))) \mid (c, c') \notin \mathbb{V}\}$ and $\bigwedge \{\forall x(\exists y V(x, y) \Rightarrow C(x)) \wedge \forall x(\exists y V(y, x) \Rightarrow C'(x))) \mid (c, c') \notin \mathbb{H}\}$, capture the adjacency constraints, thus ensuring that tiles which are not vertically (resp., horizontally) adjacent, are instead horizontally (resp., vertically) adjacent. For the reduction to proceed, we must, in addition, ensure that the grid is “closed”, i.e., that models exhibit a grid structure and no tiles fail to satisfy the adjacency contraints. This we obtain by combining φ with the sentences in (5). Such sentences express, modulo $\tau(\cdot)$, $\forall x, y(\bar{H}(x, y) \Rightarrow \neg H(x, y))$ and $\forall x, y(\bar{H}(x, y) \Rightarrow \neg H(x, y))$,

$$\begin{tikzcd}
c_{i,j+1} & & c_{i+1,j+1} \\
c_{i,j} \arrow[u, "\mathbb{V}"] \arrow[rr, "\mathbb{H}"'] & & c_{i+1,j} \arrow[u, "\mathbb{V}"']
\end{tikzcd}$$

$\mathcal{I} \models \varphi$

$$\begin{tikzcd}
c_{i,j+1} \arrow[rr, dotted, "\mathbb{H}"] & & c_{i+1,j+1} \\
c_{i,j} \arrow[u, "\mathbb{V}"] \arrow[rr, "\mathbb{H}"'] & & c_{i+1,j} \arrow[u, "\mathbb{V}"']
\end{tikzcd}$$

$\mathcal{I} \not\models \varphi$

Fig. 3. If φ is satisfied, the grid stays open, closed otherwise

Table 3. KBQA and KBSAT for the fragments of English and the positive fragments

	KBQA-UCQs	KBSAT
COP	in LSPACE	in LSPACE
COP+TV	in PTIME	in LSPACE
COP+TV+DTV	in PTIME	in LSPACE

	KBQA-CQs
COP+Rel+TV+RA^{+}	Undecidable

	KBQA-TCQs	KBSAT
COP+Rel	coNP-complete	in PTIME
COP+Rel+TV	coNP-complete	NP-complete
COP+Rel+DTV+TV	coNP-complete	NP-complete

resp., i.e., an (explicit) definition for $\bar{H}$, the formula $\xi := \forall x, y(\bar{H}(x, y) \Leftrightarrow \neg H(x, y))$: φ is false exactly when the formula $\chi := \forall x, y, z, w(H(x, y) \wedge V(x, z) \wedge V(y, w) \Rightarrow H(z, w))$, that enforces grid closure, is true, in every model of ξ. We now prove (†).

($\Leftarrow$) Let $\mathcal{I} = (\mathbb{D}_{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model of $\tau(\mathcal{S}_{\mathcal{T}}) \cup \Delta_{\mathcal{T}}$ s.t. $\mathcal{I} \not\models \varphi$ (and hence s.t., $\mathcal{I} \models \chi$). Define a mapping $f \colon \mathbb{N} \times \mathbb{N} \to \mathbb{D}_{\mathcal{I}}$ recursively as follows:

- For $i \in [0, k]$, $f(i, 0) := c_i^{\mathcal{I}}$.
- For $i \geq k$, $f(i + 1, 0) :=$ some c s.t. $(f(i, 0), c) \in H^{\mathcal{I}}$.
- For $i \geq k, j \geq 0$, $f(i + 1, j + 1) :=$ some c s.t. $(f(i, j), c) \in H^{\mathcal{I}}$.

Now, f is well-defined since *(i)* any grid point has always an $H^{\mathcal{I}}$-successor (since $\mathcal{I} \models \tau((1)))$, *(ii)* the grid is always closed (since $\mathcal{I} \not\models \varphi$) and *(iii)* $H^{\mathcal{I}}$ is non-empty (since $\mathcal{I} \models \Delta_{\mathcal{T}}$). Furthermore, by observing that $\mathcal{I}$ is modulo $\tau(\cdot)$ both a a model of (2)–(5) but not φ, one can prove by double induction on $(i, j) \in \mathbb{N} \times \mathbb{N}$ that

$$(f(i, j), f(i, j + 1)) \in H^{\mathcal{I}} \quad \text{and} \quad (f(i, j), f(i + 1, j)) \in V^{\mathcal{I}},$$

that is, f satisfies the horizontal and vertical constraints. Finally, to define the tiling $t \colon \mathbb{N} \times \mathbb{N} \to \mathbb{T}$ we put, for all $(i, j) \in \mathbb{N} \times \mathbb{N}$,

$$t(i, j) := c \quad \text{iff} \quad f(i, j) \in C^{\mathcal{I}}.$$

($\Rightarrow$) For the converse let t be a tiling for $\mathcal{T} = (\mathbb{T}, \mathbb{V}, \mathbb{H})$. We have to build a model $\mathcal{I}$ s.t. $\mathcal{I} \models \tau(\mathcal{S}_{\mathcal{T}}) \cup \Delta_{\mathcal{T}}$ and $\mathcal{I} \not\models \varphi$. Define $\mathcal{I}$ as follows:

- $\mathbb{D}_{\mathcal{I}} := \mathbb{N} \times \mathbb{N}$, $\quad c_i^{\mathcal{I}} := (i, 0)$, for $i \in [0, n]$, $\quad \bar{H}^{\mathcal{I}} := (\mathbb{D}_{\mathcal{I}} \times \mathbb{D}_{\mathcal{I}}) \setminus H^{\mathcal{I}}$,
- $C_i^{\mathcal{I}} := \{(i, 0) \in \mathbb{D}_{\mathcal{I}} \mid c_i \text{ is a } C_i \in \mathcal{F}_{\mathcal{T}}\}$, for $i \in [0, n]$,
- $H^{\mathcal{I}} := \{((i, j), (i, j + 1)) \in \mathbb{D}_{\mathcal{I}} \times \mathbb{D}_{\mathcal{I}} \mid ((i, j), (i, j + 1)) \in \mathbb{H}\}$, and
- $V^{\mathcal{I}} := \{((i, j), (i + 1, j)) \in \mathbb{D}_{\mathcal{I}} \times \mathbb{D}_{\mathcal{I}} \mid ((i, j), (i + 1, j)) \in \mathbb{V}\}$.

Clearly, $\mathcal{I}$ is the model we are looking for. □

6 Conclusions and Related Work

We have studied the semantic data complexity of Pratt and Third's fragments of English in the contexts of structured data access (KBQA) and to a lesser degree, of declaring or updating information (KBSAT). Table 3 summarizes the data complexity results presented in this paper.

Regarding data access, we can observe that as soon as function word combinations, in either the declarations alone or the declarations and queries, expressing full Boolean conjunction and negation, yielding a so-called "Boolean-closed" fragment (i.e., expressing Boolean functions), as is the case with COP+Rel are covered by the language, the task becomes intractable. Such combinations are: either *(i)* negation in subject **NP**s or *(ii)* relatives and negation in subject **NP**s and predicate **VP**s. When, in addition to these intractable constructs, transitive verbs (expressing binary relations) and anaphoric pronouns are covered, undecidability ensues. The latter observation is particularly interesting as the fragment COP+Rel+TV+RA is decidable for plain satisfiability (Pratt and Third in [17] show that this problem is NEXPTIME-complete). See Table 3. It is also interesting to note that this result strengthens Pratt's result in [15], Th. 3, stating that KBQA for the 2-variable fragment of FO and CQs is undecidable (COP+Rel+TV+RA$^+$ is strictly subsumed by the 2-variable fragment).

Also, if membership in NP of (and a fortiori for COP+Rel+TV+RA and the fragments it subsumes) was known (cf. [15], Th. 1), it turns out that the same result can be shown for restricted 3-variable fragments such as COP+Rel+TV+DTV via resolution saturations. It can also be observed that the boundary between tractability and intractability in this case lies on whether the fragment is *both* "Boolean-closed" and expresses (via transitive verbs) binary relations.

In so doing we have identified maximal and minimal fragments (and English constructs) of known (and very expressive) controlled fragments English, tailored specifically for data management tasks, such as ACE [7], for which consistency and query answering are, resp., tractable and intractable w.r.t. data complexity.

Aknowledgements. I would wish to thank Ian Pratt-Hartmann and Diego Calvanese for the discussions that originated this paper, and the anonymous reviewers, whose comments resulted in substantial improvements.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Baader, F., Calvanese, D., Nardi, D., Patel-Schneider, P., McGuinness, D.: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
3. Berger, R.: The Undecidability of the Domino Problem. Memoirs of the American Mathematical Society. American Mathematical Society, Providence (1966)
4. Bernardi, R., Calvanese, D., Thorne, C.: Expressing DL-Lite ontologies with controlled English. In: Proceedings of the 20th International Workshop on Description Logics, DL 2007 (2007)
5. Cimiano, P., Haase, P., Heizmann, J., Mantel, M., Studer, R.: Towards portable natural language interfaces to knowledge bases - The case of the ORAKEL system. Data and Knowledge Engineering 65(2), 325–354 (2008)
6. Fermüller, C.G., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 1791–1849. Elsevier - The MIT Press (2001)

7. Fuchs, N.E., Kaljurand, K., Schneider, G.: Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2006 (2005)
8. Gurevitch, Y., Grädel, E., Börger, E.: The Classical Decision Problem. Springer, Heidelberg (2001)
9. Joyner Jr., W.H.: Resolution strategies as decision procedures. Journal of the ACM 23(3), 398–417 (1976)
10. Jurafsky, D., Martin, J.: Speech and Language Processing, 2nd edn. Prentice Hall, Englewood Cliffs (2009)
11. Kaufmann, E., Bernstein, A.: How useful are natural language interfaces to the semantic web for casual end-users? In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 281–294. Springer, Heidelberg (2007)
12. Lalément, R.: Logique, réduction, résolution. Dunod (1997)
13. Minock, M., Olofsson, P., Näslund, A.: Towards building robust natural language interfaces to databases. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 187–198. Springer, Heidelberg (2008)
14. Montague, R.: Universal grammar. Theoria 36(3), 373–398 (1970)
15. Pratt-Hartmann, I.: Data complexity of the two-variable fragment with counting quantifiers. Information and Computation 207(8), 867–888 (2008)
16. Pratt-Hartmann, I.: Computational complexity in natural language. In: Handbook of Computational Linguistics and Natural Language Processing, pp. 43–73. Wiley-Blackwell (2010)
17. Pratt-Hartmann, I., Third, A.: More fragments of language. Notre Dame Journal of Formal Logic 47(2), 151–177 (2006)
18. Rich, E.: Natural-language interfaces. Computer 19(9), 39–47 (1984)
19. Robinson, A.: A machine-oriented logic based on the resolution principle. Journal of the ACM 12(1) (1965)
20. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. Journal of Intelligent Information Systems 2(3), 265–278 (1993)
21. Szymanik, J.: The computational complexity of quantified reciprocals. In: Bosch, P., Gabelaia, D., Lang, J. (eds.) TbiLLC 2007. LNCS, vol. 5422, pp. 139–152. Springer, Heidelberg (2009)
22. Thorne, C., Calvanese, D.: The data complexity of the syllogistic fragments of English. In: Proceedings of the 2009 Amsterdam Colloquium, AC 2009 (2010)
23. Vardi, M.: The complexity of relational query languages. In: Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (1982)

Strong Paraconsistency by Separating Composition and Decomposition in Classical Logic*

Peter Verdée**

Centre for Logic and Philosophy of Science,
Ghent University, Belgium
Peter.Verdee@UGent.be

Abstract. In this paper I elaborate a proof system that is able to prove all classical first order logic consequences of consistent premise sets, without proving trivial consequences of inconsistent premises (as in $A, \neg A \vdash B$). Essentially this result is obtained by formally distinguishing consequences that are the result of merely decomposing the premises into their subformulas from consequences that may be the result of also composing 'new', more complex formulas. I require that, whenever 'new' formulas are derived, they are to be preceded by a special $+$-symbol and these $+$-preceded formulas are not to be decomposed. By doing this, the proofs are separated into a decomposition phase followed by a composition phase. The proofs are recursive, axiomatizable and, as they do not trivialize inconsistent premise sets, they define a very strong non-transitive paraconsistent logic, for which I also provide an adequate semantics.

1 Introduction

Let a rule be a metalinguistic expression of the form "from $\mathsf{A}_1, \ldots, \mathsf{A}_{n-1}$ and A_n derive B, provided $\mathsf{condition}$". Instances of A_1, $\ldots \mathsf{A}_n$ will be called the local premises of (an application of) the rule and an instance of B will be called the conclusion of (an application of) the rule. In usual natural deduction proof systems for classical logic, one is able to distinguish between what I shall call *compositional* and *decompositional* rules. Compositional rules are rules of which the conclusion nor its negation occurs as a subformula in the local premises. Decompositional rules are rules that are not compositional. Examples of typical compositional rules include Addition (from 'A' derive '$A \vee B$'), Adjunction (from 'A' and 'B' derive '$A \wedge B$'), Existential Generalization (from '$A(\alpha)$' derive '$\exists \beta A(\beta)$', where β is a variable) and Introduction of the Identity (derive $\alpha = \alpha$). Typical decompositional rules are Conjunction Elimination (from '$A \wedge B$' derive 'A'), Universal Instantiation (from '$\forall \alpha A(\alpha)$' derive '$A(\beta)$'), Modus Ponens (from

[*] I am indebted to the referees for providing useful comments on a former draft and for pointing my attention to some interesting related literature.

[**] Peter Verdée is a post-doctoral fellow of the Fund for Scientific Research – Flanders.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 272–292, 2011.

'$A \supset B$' and 'A' derive 'B') and Modus Tollens (from '$A \supset B$' and '$\neg B$', derive '$\neg A$').

How can we interpret such a distinction? It is clarifying to look at this distinction from a information-theoretical point of view. We could consider rules as means to obtain more information from already derived information. Every step of a prove governed by rules reveals or discovers more information from the premises of the proof or from the laws of pure logic (in case of theorems)[1]. This process of revealing information goes in two directions. Decompositional rules reveal the information that is already fully present in the local premises. They make this information available for further proof steps. Compositional rules on the other hand somehow construct new information: they construct new (true) strings by bringing new symbols into the proof (in the case of Addition) or by combining already available strings and symbols into new more complex strings. The compositional steps of interesting proofs create new information about the premises, while the decompositional steps only analyze the premises. Compare it to a toy house built with Lego blocks. The decompositional steps correspond to the breaking down of the house into smaller parts, without doing anything new. The compositional steps correspond to building up the smaller parts into new creative houses or other constructions.

This distinction has an interesting computational aspect. One needs no goal directed procedure to apply all possible decompositional rules to a (finite) set of premises. This can be done in linear time, because the resulting formulas can only become smaller. Of course this is not the case for the compositional rules. One is, for example, able to apply the rule 'from A derive $A \wedge A$' infinitely many times starting from one premise without ever obtaining the same formula. Hence, in order to have any chance at ever finding an interesting result by applying compositional rules, one needs a goal directed procedure, i.e. an algorithm that determines which rules to apply in order to be able to derive interesting formulas (goals).

My aim in this paper is threefold: first, I shall show that it is possible to devise a proof system for classical first order logic (henceforth called **CL**) in which one can properly distinguish the compositional from the decompositional rules. This might seem evident, but remember that fitch style proofs for example use conditional proofs. The status of the rule that allows one to derive an implication from a conditional proof is obviously a compositional rule. However, the conditional proof itself might also use decomposition, while the results of decomposition steps in conditional proofs definitely are not decompositions of the premises of the main proof. So, what would their status be? A usual axiom system has another problem: all steps, apart from Modus Ponens and the introduction of premises,

[1] According to the traditional view on information, logical deduction cannot provide new information. This position however is not anymore considered tenable by various contemporary scholars (cf. [7] and [11]). A more epistemic perspective on the matter, taking into account the fact that realistic agents do not have logical omniscience, calls for a less extreme position. Here I look at information from an intuitive epistemic stance. A more formal elaboration of my stance on information in deductive reasoning contexts is forthcoming.

are compositional (or decompositional, depending on whether one sees axioms as available information). Obviously this makes the distinction rather useless. The proof system in this paper makes a clear distinction between compositional and decompositional steps. This distinction is even present at the object level: compositional steps result in formulas preceded by a special +-symbol. The logic **C**, which I shall present here will formalize this proof system. The logic has a straight forward axiomatization and is therefore a usual Tarski logic: compact, monotonic, reflexive and transitive.

Secondly, I want to show that, using this proof system, one only needs one phase of decompositional steps followed by one phase of compositional steps in order to obtain the full consistent fragment of classical logic. This is implemented by requiring that, whenever 'new', composed formulas are derived, they are to be preceded by a special +-symbol and these +-preceded formulas are not to be decomposed. The decomposition phase, i.e. the phase in which formulas are only decomposed and no +-preceded formulas are obtained, could be interpreted as the analysis of the information present in the premises into the essential parts needed for the decomposition phase, where one is able to derive new, composed formulas. For the propositional level, the first phase more or less comes to analyzing the premises into a normal form. However, at the predicative level this is more complex.

Thirdly, and this is the main result, I want to show that one obtains a very nice paraconsistent logic called $\mathbf{CL}^-$ by separating composition and decomposition. This first order logic is paraconsistent, in the sense that inconsistent premise sets do not lead to a trivial consequence set but the only thing one has to give up by moving to this special paraconsistent logic is transitivity (sometimes also called the 'Cut'-rule, i.e. if $\Gamma \vdash C$ for all $C \in \Delta$ and $\Gamma \cup \Delta \vdash B$ then $\Gamma \vdash B$). The logic is computationally not more complex than **CL** and has an elegant proof system. Moveover, and most importantly, everything useful that is derivable using **CL**, can also be derived using **C**. This logic defines a consequence relation that is only weaker than the **CL**-consequence relation in the following two aspects: (i) the trivial consequences of inconsistent premises are lost, but I suppose nobody will see this as a disadvantage and (ii) the consequence relation is no longer transitive. This last property might be considered as a disadvantage, but is solved by the fact that the logic **C** itself is transitive. Although **C** defines a weaker paraconsistent consequence relation than $\mathbf{CL}^-$ in the usual **CL** language, the logics **C** and $\mathbf{CL}^-$ have essentially the same proof theory and every reasoning process explicated by $\mathbf{CL}^-$ can also be explicated by the logic **C**.

The results in this paper are strongly inspired by important work by Batens and Provijn on so called *goal directed* or *prospective* proofs (cf. [1], [3], [4] and [12]). The proofs they present are essentially procedural and goal dependent, i.e. the correctness of a proof does not only depend on the premises of the proof but also on the goal of the proof (the formula that is to be derived). The propositional version of the logic $\mathbf{CL}^-$ is an unexpected result of their goal directed proof system. It is my aim to (i) generalize their results on $\mathbf{CL}^-$ to the first order level and (ii) to translate their proofs to usual non goal dependent, non procedural proofs. This has the important advantage that one can accumulate and reuse conclusions from a set of premises, i.e. old results obtained in proofs for some goal A can be reused in proofs for a different goal B. This way I am able to reintroduce transitivity. Although I give up goal dependence, I shall show that my proofs can still very easily be turned into goal directed proofs, by adding some heuristic restrictions to the rules of the logic.

The logic presented in this paper is also related to Besnard and Hunter's quasi-classical logic (cf. [5], [9], and [10]) and Craig's Linear Reasoning (cf. [6]), which is inspired by Gentzen's famous cut-elimination theorem (cf. [8]). Both approaches separate compositional from decompositional steps in classical logic in a very similar way (apart from some details, the distinction is identical to the one I make). Craig, however, did not relate his results to paraconsistency, neither did he give a semantics for his proof system. Besnard and Hunter's fascinating QC logic is quite similar to the logic $\mathbf{CL}^{-}$ (apart from the fact that their consequence relation (in general) does not validate **CL**-theorems), but lacks some of the attractive properties of the here presented logic **C**, viz. (1) they do not present an axiomatization, (2) they do not introduce a symbol (like the +-symbol) to express the difference between decomposition and composition at object level, (3) their proofs do not show the attractive relevant logic interpretation that I shall explain in section 7, and (4) the semantic construction I shall present explicitly separates compositional and decomposition steps by means of gluts or gaps for all formulas, whereas their semantics is only an adequate semantics for the consequence relation **Q** (not for individual proof steps).

2 Language and Proof Theory

I start by presenting the formal languages that occur in this paper.

Let $\mathcal{L}$ be the language of **CL** with the logical symbols $\neg$, $\supset$, $\exists$, and $=$ (but without function symbols); $\mathcal{C}$ is the set of (letters for) individual constants, $\mathcal{V}$ the set of individual variables, $\mathcal{S}$ is the set of sentential letters, and $\mathcal{P}^r$ the set of predicates of rank $r \geq 0$—predicates of rank 0 will function as sentential letters. The members of $\mathcal{P}^r$ will be $P^r, Q^r, R^r, P^r_1, \ldots$. Let $\mathcal{F}$ denote the set of (possibly open) formulas of $\mathcal{L}$ and let $\mathcal{W}$ denote the set of closed formulas of $\mathcal{L}$. Let ${}^{\neg}\mathcal{P}^r$ be the set of the members of $\mathcal{P}^r$ to which a $\neg$-symbol is added as a subscript, i.e. the set $\{P^r_{\neg}, Q^r_{\neg}, R^r_{\neg}, P^r_{1\neg}, \ldots\}$ and let ${}^{\neg}\mathcal{S}$ be the set of the members of $\mathcal{S}$ to which a $\neg$-symbol is added as a subscript, i.e. the set $\{p_{\neg}, q_{\neg}, r_{\neg}, p_{1\neg}, \ldots\}$.

But the actual language of the logic **C** contains an extra +-symbol. Let $\mathcal{L}^{+}$ be exactly like $\mathcal{L}$, apart from the fact that the unary symbol $+$ is added to the language. This symbol should only occur in front of formulas and should not be nested. $\mathcal{W}^{+} = \{+A \mid A \in \mathcal{W}\} \cup \mathcal{W}$. $\mathcal{F}^{+}$ is defined analogously.

$\mathbb{P}$ denotes the set of all members of $\mathcal{W}$ in which no logical symbols (not even identity) occur and $\mathbb{I} = \{\alpha = \beta \mid \alpha, \beta \in \mathcal{C}\}$. $\mathbb{P}^{\neg} = \mathbb{P} \cup \{\neg A \mid A \in \mathbb{P}\}$ and $\mathbb{P}^{+} = \mathbb{P} \cup \{+A \mid A \in \mathbb{P}\}$. $\mathbb{I}^{+}$ and $\mathbb{I}^{\neg}$ are defined analogously.

Let us now turn to the description of the proof system. In order to facilitate this, I introduce the abbreviations $*$ and $\ddagger$: where $A \in \mathcal{W}$, $*\neg A = A$, $*A = \neg A$ if A is not of the form $\neg B$, $\ddagger + A =_{\text{df}} *A$, and $\ddagger A =_{\text{df}} + * A$.

Lines of proofs do not contain just one formula, but a finite set of formulas between $\llbracket$ and $\rrbracket$. I use the sloppy notation $\llbracket A_1, \ldots, A_n, \Delta_1, \ldots, \Delta_k, B_1, \ldots, B_m \rrbracket$, where $\{A_1, \ldots, A_n\}$ and $\{B_1, \ldots, B_n\}$ are possibly empty, to denote the line $\llbracket \{A_1, \ldots, A_n, B_1, \ldots, B_m\} \cup \Delta_1 \cup \ldots \cup \Delta_k \rrbracket$. The intuitive meaning of such lines is that at least one of the formulas in the set between $\llbracket$ and $\rrbracket$ on a derived line is derivable. But note that this interpretation is formally not entirely correct: the lines are actually a lot stronger than this, and behave like contraposable relevant conditionals, cf. section 7.

I start by presenting an axiomatic proof system with 9 axiom schemata and 3 rules[2].

AS1 $\llbracket *A, +A \rrbracket$
AS2 $\llbracket + * A, +A \rrbracket$
AS3 $\llbracket A, B, +\neg(A \vee B) \rrbracket$
AS4 $\llbracket *B, +(A \vee B) \rrbracket$
AS5 $\llbracket +\alpha = \alpha \rrbracket$
AS6 $\llbracket *A(\alpha), +\exists x A(x) \rrbracket$
AS7 $\llbracket +A(\alpha), + * A(\beta), +\neg\alpha = \beta \rrbracket$
AS8 $\llbracket +\neg\alpha = \beta, +\neg\beta = \gamma, \alpha = \gamma \rrbracket$
AS9 $\llbracket +\neg\alpha = \beta, \beta = \alpha \rrbracket$
PREM if $A \in \Gamma$, derive $\llbracket A \rrbracket$
TRANS from $\llbracket A, \Delta_1 \rrbracket$ and $\llbracket \ddagger A, \Delta_2 \rrbracket$ derive $\llbracket \Delta_1 \cup \Delta_2 \rrbracket$
UG if $\beta \in \mathcal{C}$ does not occur in $\Delta \cup \Gamma \cup \{A(\alpha)\}$, from $\llbracket \Delta, +A(\beta) \rrbracket$ derive $\llbracket \Delta, +\neg\exists\alpha * A(\alpha) \rrbracket$

Definition 1. *A* **C***-proof from a set of premises Γ is a list of lines of the form $\llbracket \Delta \rrbracket$ where $\Delta \subset \mathcal{W}^+$ is finite and each line is an instance of one of the axiom schemata AS1-9 or the result of an application of one of the rules TRANS, PREM or UG.*

Definition 2. *Where $\Gamma \cup \{A\} \subset \mathcal{W}^+$, $\Gamma \vdash_{\mathbf{C}} A$ iff there is a* **C***-proof from premises Γ in which $\llbracket A \rrbracket$ occurs on a line. Where $\Gamma \cup \Delta \subset \mathcal{W}^+$, $\Gamma \vdash_{\mathbf{C}} \llbracket \Delta \rrbracket$ iff there is a* **C***-proof from premises Γ in which $\llbracket \Delta \rrbracket$ occurs on a line.*

This completes the definition of the proof system **C**. I now introduce a more intuitive rule system called **C2** (with many redundant, derivable rules). It inherits the 3 rules of **C**, but 15 rules are added.

R1 derive $\llbracket *A, +A \rrbracket$
R2 from $\llbracket A, \Delta \rrbracket$ derive $\llbracket +A, \Delta \rrbracket$
R3 from $\llbracket + * A, + * B, \Delta \rrbracket$ derive $\llbracket +\neg(A \vee B), \Delta \rrbracket$
R4 from $\llbracket A \vee B, \Delta \rrbracket$ derive $\llbracket A, B, \Delta \rrbracket$
R5 from $\llbracket +A, \Delta \rrbracket$ derive $\llbracket +(A \vee B), \Delta \rrbracket$
R6 from $\llbracket +B, \Delta \rrbracket$ derive $\llbracket +(A \vee B), \Delta \rrbracket$

[2] Concerning the rule UG: if all $\beta \in \mathcal{C}$ already occur in Γ, one may introduce dummy constants in **C**-proofs, but the formulas that are the conclusions of the proof should not contain these dummy constants.

R7 from $\llbracket \neg(A \vee B), \Delta \rrbracket$ derive $\llbracket *A, \Delta \rrbracket$
R8 from $\llbracket \neg(A \vee B), \Delta \rrbracket$ derive $\llbracket *B, \Delta \rrbracket$
R9 derive $\llbracket +\alpha = \alpha \rrbracket$
R10 $\llbracket +\neg\alpha = \beta, +\neg\beta = \gamma, \alpha = \gamma \rrbracket$
R11 $\llbracket +\neg\alpha = \beta, \beta = \alpha \rrbracket$
R12 from $\llbracket +A(\alpha), \Delta \rrbracket$ derive $\llbracket +\exists x A(x), \Delta \rrbracket$
R13 from $\llbracket \neg\exists x A(x), \Delta \rrbracket$ derive $\llbracket *A(\alpha), \Delta \rrbracket$
R14 from $\llbracket A(\alpha), \Delta \rrbracket$ and $\llbracket *A(\beta), \Delta \rrbracket$ derive $\llbracket +\neg\alpha = \beta, \Delta \rrbracket$
R15 from $\llbracket \alpha = \beta, \Delta \rrbracket$ and $\llbracket A(\alpha), \Delta \rrbracket$ derive $\llbracket +A(\beta), \Delta \rrbracket$

Definition 3. *A* **C2***-proof from a set of premises Γ is a list of lines of the form $\llbracket \Delta \rrbracket$ where $\Delta \subset \mathcal{W}^+$ and each line is the result of an application of one of the rules TRANS, PREM or UG or R1-R15.*

Definition 4. *$\Gamma \vdash_{\mathbf{C2}} A$ iff there is a* **C***-proof from premises Γ in which $\llbracket A \rrbracket$ occurs on a line. $\Gamma \vdash_{\mathbf{C2}} \llbracket \Delta \rrbracket$ iff there is a* **C***-proof from premises Γ in which $\llbracket \Delta \rrbracket$ occurs on a line.*

The proof systems **C** and **C2** lead to exactly the same consequence relation. This is stated in the following theorem, which can easily be proved by showing that all **C**-axioms can be derived using **C2** rules and vice versa.

Theorem 1. *$\Gamma \vdash_{\mathbf{C}} \llbracket \Delta \rrbracket$ iff $\Gamma \vdash_{\mathbf{C2}} \llbracket \Delta \rrbracket$.*

C2 has the advantage of yielding more natural proofs and explicitly showing the difference between composition and decomposition, while **C** is metatheoretically more elegant.

In both **C** and **C2** one is able to define the other usual connectives in the usual way, i.e. $A \supset B =_{\text{df}} *A \vee B$, $A \wedge B =_{\text{df}} \neg(*A \vee *B)$, $A \equiv B =_{\text{df}} (A \supset B) \wedge (B \supset A)$, and $\forall\alpha A(\alpha) =_{\text{df}} \neg\exists\alpha * A(\alpha)$. In the example proofs, I shall use these defined symbols as abbreviations for their definientia.

Where **L** is a logic, $\mathcal{W}'$ is the language of **L**, and $\Gamma \subseteq \mathcal{W}'$, let $Cn_{\mathbf{L}}(\Gamma) =_{\text{df}} \{A \in \mathcal{W}' \mid \Gamma \vdash_{\mathbf{L}} A\}$. The following theorem will be proved later (in section 7 as theorem 9), but **C** is a logic that is derived from **CL** and the **C**-proofs are meant to prove **CL**-consequences, so this theorem is essential to understand the function of **C**-proofs; it says that one can prove every **CL**-consequence of every premise set $\Gamma \subset \mathcal{W}$ with a **C**-proof, as far as $Cn_{\mathbf{CL}}(\Gamma)$ is not trivial[3].

Theorem 2. *If there is some C such that $\Gamma \nvdash_{\mathbf{CL}} C$, then $\Gamma \vdash_{\mathbf{C}} +A$ iff $\Gamma \vdash_{\mathbf{CL}} A$.*

Although our proofs are strong enough to prove all useful **CL**-consequences, they respect the requirements for the behavior of the +-symbol: (i) every time a new, more complex formula is derived, it is preceded by a +-symbol (for example $\{Pa\} \vdash_{\mathbf{C}} +\exists x Px$, $\vdash_{\mathbf{C}} +a = a$, $\{Pa, Pb\} \vdash_{\mathbf{C}} +(Pa \wedge Pb)$ and $\{Pa \vee Pb\} \vdash_{\mathbf{C}} +(Pb \vee Pa)$ but $\{Pa\} \nvdash_{\mathbf{C}} \exists x Px$, $\nvdash_{\mathbf{C}} a = a$, $\{Pa, Pb\} \nvdash_{\mathbf{C}} Pa \wedge Pb$

[3] Remark that this is not a weakness: if $Cn_{\mathbf{CL}}(\Gamma)$ is trivial, proving **CL**-consequences of Γ is quite useless anyway.

and $\{Pa \vee Pb\} \nvdash_{\mathbf{C}} Pb \vee Pa$) and on the other hand (ii) a formula which is preceded by a +-symbol is never decomposed ($\{+\forall x(Px \wedge Rx)\} \nvdash_{\mathbf{C}} +Pa$ and $\{+(Pa \supset Pb), Pa\} \nvdash_{\mathbf{C}} Pb$). The only case where the requirements for the +-symbol are slightly loosened is the case of identity. In order to obtain an elegant metatheory, I had to allow $\{a = b, b = c\} \vdash_{\mathbf{C}} a = c$, $\{a = b\} \vdash_{\mathbf{C}} b = a$ and $\{a = b\} \vdash_{\mathbf{C}} a = a$, although, strictly speaking, the ideas from section 1 would have required that these conclusions would only be allowed when preceded by a +-symbol, as e.g. $a = a$ or $b = a$ or their negations are not subformulas of $a = b$. Remark, however, that this is not a severe violation of the principles. Unlike in the case of e.g. what is sometimes called the stuttering rule ($A \vdash A \wedge A$), which is not valid in **C**, these identity transitivity and symmetry rules can never result in infinitely many +-free consequences of finitely many premises. For example the only +-free **C**-consequences of $\{a = b, b = c\}$ are $\{a = a, a = b, a = c, b = a, b = b, b = c, c = a, c = b, c = c\}$. As a consequence, these identity rules might also be seen as rules that result in a special kind of decomposition steps, rather than composition steps. Do not confuse this special case with normal identity cases, e.g. $\{Pa, a = b\} \nvdash_{\mathbf{C}} Pb$ and $\{Pa\} \nvdash_{\mathbf{C}} a = a$, as the normal cases are perfectly in line with the requirements.

Finally, the attention of the reader should be pointed at the paraconsistency of **C**, i.e. for every $A \in \mathcal{W}^+$, there is a $B \in \mathcal{W}^+$ such that $A, \neg A \nvdash_{\mathbf{A}} B$. One can prove this by means of the paraconsistency of the semantics of **C** defined in Section 4 plus the soundness of the proof theory of **C** with respect to its semantics (cf. Section 5). However, to get a hint of how explosion is avoided, observe that there is no immediate Ex Falso Quodlibet rule in **C**. Also, the alternative way to derive explosion always involves composing a new formula which is later decomposed (for example by applying Addition and Disjunctive Syllogisme to an inconsistency: in case we have p and $\neg p$, simply apply $p/p \vee q$ (Addition) and $\neg p, p \vee q/q$ (Disjunctive Syllogism) — $p \vee q$ as a newly composed formula which is decomposed later). Precisely this is blocked in **C** whence this blockage makes explosion impossible and the logic paraconsistent.

3 Examples

C-proof for $a = b \vdash_{\mathbf{C}} a = a$

1	$[\![a = b]\!]$	PREM
2	$[\![+\neg a = b, +\neg b = a, a = a]\!]$	AS8
3	$[\![+\neg b = a, a = a]\!]$	TRANS; 1,2
4	$[\![+\neg a = b, b = a]\!]$	AS9
5	$[\![b = a]\!]$	TRANS; 1,4
6	$[\![a = a]\!]$	TRANS; 3,5

C2-proof for $\exists x \forall y Pxy \vdash_{\mathbf{C}} \forall y \exists x Pxy$

1	$[\![\exists x \forall y Pxy]\!]$	PREM
2	$[\![\forall y Pay, + * \forall y Pay]\!]$	R1
3	$[\![Pab, + * \forall y Pay]\!]$	R13; 2
4	$[\![+Pab, + * \forall y Pay]\!]$	R2; 3
5	$[\![+\exists x Pxb, + * \forall y Pay]\!]$	R12; 4

6	$\llbracket +\forall y\exists x Pxy, + * \forall y Pay \rrbracket$	UG; 5
7	$\llbracket +\forall y\exists x Pxy, + * \exists x\forall x Pxy \rrbracket$	UG; 6
8	$\llbracket +\forall y\exists x Pxy \rrbracket$	TRANS; 1,7

C2-proof for $\vdash_{\mathbf{C}} +\forall x\exists y(((Px \wedge \neg Rx) \wedge (\neg Py \vee \neg Ry)) \supset \neg x = y)$
(in the following proof, let X abbreviate $+(((Pa \wedge \neg Ra) \wedge (\neg Pb \vee \neg Rb)) \supset \neg a = b)$)

1	$\llbracket \neg(((Pa \wedge \neg Ra) \wedge (\neg Pb \vee Rb)) \supset \neg a = b), X \rrbracket$	R1
2	$\llbracket (Pa \wedge \neg Ra) \wedge (\neg Pb \vee Rb), X \rrbracket$	R7; 1
3	$\llbracket Pa \wedge \neg Ra, X \rrbracket$	R7; 2
4	$\llbracket Pa, X \rrbracket$	R7; 3
5	$\llbracket \neg Ra, X \rrbracket$	R8; 3
6	$\llbracket \neg Pb \vee Rb, X \rrbracket$	R8; 2
7	$\llbracket \neg Pb, Rb, X \rrbracket$	R4; 6
8	$\llbracket \neg a = b, Rb, X \rrbracket$	R10; 4,7
9	$\llbracket \neg a = b, X \rrbracket$	R10; 5,8
10	$\llbracket a = b, X \rrbracket$	R8; 1
11	$\llbracket +a = b, X \rrbracket$	R2; 10
12	$\llbracket X \rrbracket$	TRANS; 9,11
13	$\llbracket +\exists y(((Pa \wedge \neg Ra) \wedge (\neg Py \vee \neg Ry)) \supset \neg a = y) \rrbracket$	R12; 12
14	$\llbracket +\forall x\exists y(((Px \wedge \neg Rx) \wedge (\neg Py \vee \neg Ry)) \supset \neg x = y) \rrbracket$	UG; 11

C2-proof for $\forall x(\neg Px \vee (Qx \wedge Rx)), Pa \vdash_{\mathbf{C}} +\exists y(Qy \vee Sy)$

1	$\llbracket \forall x(\neg Px \vee (Qx \wedge Rx)) \rrbracket$	PREM
2	$\llbracket \neg Pa \vee (Qa \wedge Ra) \rrbracket$	R13; 1
3	$\llbracket Pa \rrbracket$	PREM
4	$\llbracket \neg Pa, Qa \wedge Ra \rrbracket$	R4; 2
5	$\llbracket +\neg Pa, Qa \wedge Ra \rrbracket$	R2; 4
6	$\llbracket Qa \wedge Ra \rrbracket$	TRANS; 5
7	$\llbracket Qa \rrbracket$	R7; 6
8	$\llbracket +Qa \rrbracket$	R2; 7
9	$\llbracket +(Qa \vee Sa) \rrbracket$	R5; 8
10	$\llbracket +\exists y(Qy \vee Sy) \rrbracket$	R12; 9

4 Semantics

Because this proof system (although it is a proof system for (consistent) **CL**) shows, thanks to its +-symbol and its paraconsistency, behavior that is quite different from **CL**, it should not come as a surprise that one needs a completely different semantics for **C**.

We shall use pseudo-constants to present the semantics as efficiently as possible. They do not occur in the actual object language (in premises, proofs or conclusions) but only in the semantics. The set $\mathcal{O}$ is the set of pseudo-constants. The language $\mathcal{L}_o$ is the language $\mathcal{L}$ enriched with these constants. The sets $\mathcal{F}_o$,$\mathcal{W}_o$,$\mathbb{P}_o$, $\mathbb{I}_o$,$\mathcal{F}^+_o$,$\mathcal{W}^+_o$,$\mathbb{P}^+_o$, $\mathbb{I}^+_o$, $\mathbb{P}^-_o$, and $\mathbb{I}^-_o$ are identical to the respective sets without subscript o, except for the replacement of $\mathcal{C}$ by $\mathcal{C} \cup \mathcal{O}$ in their definitions.

The semantics I am about to present is perfectly deterministic but has the unusual property that the assignment assigns (unrelated) truth values to all formulas (not only the primitive ones). This technical choice is made in order to ensure that all formulas without a +-symbol show a logical gap (the normal truth condition for the formula might be true, whilst the formula itself is false) and all formulas with a $+$-symbol show a logical glut (the normal truth condition for the formula might be false, whilst the formula itself is true). In this way we obtain that our requirements about $+$-symbols are satisfied.

A **C**-model M is a triple $\langle v, w, D\rangle$, where D (the domain) is some set of objects, $w\colon \mathcal{W}_o^+ \to \{0,1\}$ is an assignment function which assigns a truth value to every formula, without respecting the structure of the formula, and v is an assignment function that has the following properties:

(i) $v\colon \mathcal{C} \cup \mathcal{O} \to D$ (with $D = \{v(\alpha) \mid \alpha \in \mathcal{C} \cup \mathcal{O}\}$)
(ii) $v\colon \mathcal{S} \cup {}^{\neg}\mathcal{S} \to \{0,1\}$
(iii) $v\colon \mathcal{P}^r \cup {}^{\neg}\mathcal{P}^r \to \wp(D^{(r)})$ (for every $r \geq 1$).
(iv) $v\colon \{\neg\cdot{=}\cdot, \cdot{=}\cdot, +\neg\cdot{=}\cdot\} \to \wp(D^{(2)})$

Let $I_v(A)$, where v is an assignment function with the properties mentioned above and A is an atomic formula, be defined by '$I_v(\pi^r\alpha_1 \ldots \alpha_r) =_{df} \langle v(\alpha_1), \ldots, v(\alpha_r)\rangle \in v(\pi^r)$', '$I_v(\neg\pi^r\alpha_1 \ldots \alpha_r) =_{df} \langle v(\alpha_1), \ldots, v(\alpha_r)\rangle \in v(\pi^r_\neg)$', '$I_v(\sigma) =_{df} v(\sigma) = 1$', '$I_v(\neg\sigma) =_{df} v(\sigma_\neg) = 1$', '$I_v(\alpha = \beta) = \langle v(\alpha), v(\beta)\rangle \in v(\cdot = \cdot)$', '$I_v(\neg\alpha = \beta) = \langle v(\alpha), v(\beta)\rangle \in v(\neg\cdot = \cdot)$', and '$I_v(+\neg\alpha = \beta) = \langle v(\alpha), v(\beta)\rangle \in v(+\neg\cdot = \cdot)$'. Let $T_M(\beta)$, where M is a model $\langle v, w, D\rangle$ and $\beta \in D$, abbreviate the semantic statement 'either $\langle \beta, \beta\rangle \in v(\neg\cdot{=}\cdot)$ or there is a r-ary predicate π and $\alpha_1, \ldots, \alpha_{r-1} \in D$, such that $\langle \alpha_1, \ldots, \alpha_i, \beta, \alpha_{i+1}, \ldots, \alpha_{r-1}\rangle \in v(\pi^r) \cap v(\pi^r_\neg)$'.

Definition 5. *The function $\ddagger$ which maps metalinguistic semantic expressions to metalinguistic semantic expressions is recursively defined by means of the following clauses (between quotation marks)—note that I write square brackets to avoid confusion between function brackets and brackets in semantic expressions: "$\ddagger[\mathsf{A}$ or $\mathsf{B}] =_{df} \ddagger\mathsf{A}$ and $\ddagger\mathsf{B}$", "$\ddagger[\mathsf{A}$ and $\mathsf{B}] =_{df} \ddagger\mathsf{A}$ or $\ddagger\mathsf{B}$", "$\ddagger[$for all $\beta \in \mathcal{C} \cup \mathcal{O}, \mathsf{A}] =_{df}$ there exists a $\beta \in \mathcal{C} \cup \mathcal{O}$ such that $\ddagger\mathsf{A}$", "$\ddagger[$there exists a $\beta \in \mathcal{C} \cup \mathcal{O}$ such that $\mathsf{A}] =_{df}$ for all $\beta \in \mathcal{C} \cup \mathcal{O}$, $\ddagger\mathsf{A}$", "$\ddagger[w(A) = 1] =_{df} w(A) = 0$ or $w(\ddagger A) = 1$", "$\ddagger[I_v(A)] =_{df}$ not $I_v(A)$ or $I_v(\neg A)$" where $A \in \mathbb{P}$, "$\ddagger[I_v(\alpha = \beta)] =_{df}$ not $I_v(\alpha = \beta)$ or $I_v(+\neg\alpha = \beta)$", "$\ddagger[$not $I_v(A)$ or $I_v(\neg A)] =_{df} I_v(A)$" where $A \in \mathbb{P}$, "$\ddagger[v(\alpha) \neq v(\beta)$ or $T_M(v(\alpha))] =_{df} v(\alpha) = v(\beta)$", "$\ddagger[v(\alpha) = v(\beta)] =_{df} v(\alpha) \neq v(\beta)$ or $T_M(v(\alpha))$" and finally "$\ddagger[v_M(A) = i] =_{df} v_M(\ddagger A) = i$", where $i \in \{0,1\}$.*

The valuation function $v_M\colon \mathcal{W}_o^+ \to \{0,1\}$, where $M = \langle v, w, D\rangle$ is a **C**-model, is defined by the following clauses.

S1 where $A \in \mathbb{P}$, $v_M(A) = 1$ iff $w(A) = 1$ and $I_v(A)$
S2 where $A \in \mathbb{P}$, $v_M(\neg A) = 1$ iff $w(\neg A) = 1$ and (not $I_v(A)$ or $I_v(\neg A)$)
S3 $v_M(\neg\neg A) = 1$ iff $w(\neg\neg A) = 1$ and $v_M(A) = 1$
S4 $v_M(\neg(A \vee B)) = 1$ iff $w(\neg(A \vee B)) = 1$ and $v_M(*A) = 1$ and $v_M(*B) = 1$
S5 $v_M(A \vee B) = 1$ iff $w(A \vee B) = 1$ and ($v_M(+*A) = 0$ or $v_M(B) = 1$) and ($v_M(+*B) = 0$ or $v_M(A) = 1$)
S6 $v_M(\neg\exists\alpha A(\alpha)) = 1$ iff $w(\neg\exists\alpha A(\alpha)) = 1$ and for all $\beta \in \mathcal{C} \cup \mathcal{O}$, $v_M(\neg A(\beta)) = 1$

S7 $v_M(\exists\alpha A(\alpha)) = 1$ iff $w(\exists\alpha A(\alpha)) = 1$ and there exists a $\beta \in \mathcal{C} \cup \mathcal{O}$, such that $v_M(A(\beta)) = 1$
S8 $v_M(\neg\alpha = \beta) = 1$ iff $w(\neg\alpha = \beta) = 1$ and $(v(\alpha) \neq v(\beta)$ or $T_M(v(\alpha)))$
S9 $v_M(\alpha = \beta) = 1$ iff $(I_v(\alpha = \beta)$ or $w(\beta = \beta) = 1)$ and $v(\alpha) = v(\beta)$
S10 $v_M(+A) = 1$ iff $\ddagger[v_M(*A) = 1]$

Clauses S2, S8 and S10 deserve a little extra attention. Clause S2 and S8 introduce another type of gluts than the ones introduced to fulfil the requirement for the +-symbol. This glut is a typical negation glut and ensures the paraconsistency of the logic. Remark that the logic needs to be paraconsistent as we obviously do not want to decompose formulas A and $\neg A$ into an arbitrary formula B. Clause S10 abbreviates all clauses for formulas of the form $+A$. They are generated by $\ddagger$-transforming the clause for $*A$ in line with Definition 5.

Given these criteria for being a **C** valuation function v_M we can obtain the following properties (for every $A \in \mathcal{W}_o^+$ and every **C**-model M) by a straight forward mathematical induction on the complexity of A.

Lemma 1. *1. if $v_M(A) = 1$ then $v_M(+A) = 1$*
2. $v_M(A) = 1$ or $v_M(\ddagger A) = 1$

Finally we have all the means to determine when a **C**-model satisfies a formula and to define semantic consequence.

Definition 6. *Satisfaction.*
*Where $A \in \mathcal{W}^+$ and M is a **C**-model, $M \models A$ iff $v_M(A) = 1$.*

Definition 7. *Semantic **C**-consequence.*
Where $\Gamma \cup \{A\} \subset \mathcal{W}^+$, $\Gamma \vDash_{\mathbf{C}} A$ iff $M \models A$ for every model M such that $M \models B$ for all $B \in \Gamma$.

5 Soundness, Completeness and Other Important Properties

We start by some important properties of the logic **C**. The next interesting Lemma might be seen as an alternative for the usual deduction theorem. It is proven in the Appendix of this paper.

Lemma 2. *If $\Delta' \subset \Delta$ and $\Gamma \cup \{\ddagger B | B \in \Delta'\} \vdash_{\mathbf{C}} [\![\Delta - \Delta']\!]$, then there is a $\Theta \subset \{\ddagger B | B \in \Delta'\}$ such that $\Gamma \cup \Theta \vdash_{\mathbf{C}} [\![\Delta - \Delta']\!]$ or $\Gamma \vdash_{\mathbf{C}} [\![\Delta]\!]$*

This lemma has some interesting corollaries.

Corollary 1. *If $\Gamma \cup A \vdash_{\mathbf{C}} B$ then $\Gamma \vdash_{\mathbf{C}} B$ or $\Gamma \vdash_{\mathbf{C}} [\![\ddagger A, B]\!]$.*

Corollary 2. *If $\Gamma \cup A \vdash_{\mathbf{C}} B$ then $\Gamma \vdash_{\mathbf{C}} B$ or $\Gamma \cup \{\ddagger B\} \vdash_{\mathbf{C}} \ddagger A$.*

Corollary 3. *If $\Gamma \cup \{A\} \vdash_{\mathbf{C}} B$ and $\Gamma \cup \Delta \cup \{\ddagger A\} \vdash_{\mathbf{C}} B$, then $\Gamma \cup \Delta \vdash_{\mathbf{C}} B$.*

The next lemma immediately entails the soundness theorem for **C**.

Lemma 3. *If $\Gamma \vdash_{\mathbf{C}} [\![\Delta]\!]$ then for all $A \in \Delta$, $\Gamma \cup \{\ddagger B \mid B \in \Delta - \{A\}\} \vDash_{\mathbf{C}} A$.*

Proving Lemma 3 is just a matter of letting every line of every proof correspond to its appropriate semantic **C**-consequence and proving that every **C**-rule corresponds to a correct fact about semantic **C**-consequences. This is quite straight forward except maybe for the fact that one should be aware of the fact that $[\![A, A]\!]$ is identical to $[\![A]\!]$. The semantic fact that correspond to this is that $\Gamma \cup \{\ddagger A\} \vDash_{\mathbf{C}} A$ iff $\Gamma \vDash_{\mathbf{C}} A$, which is warranted by Lemma 1.

Theorem 3. *Soundness of* **C**.
For every $\Gamma \cup \{A\} \subseteq \mathcal{W}^+$, if $\Gamma \vdash_{\mathbf{C}} A$ then $\Gamma \vDash_{\mathbf{C}} A$

The following completeness theorem is proved in the appendix of this paper.

Theorem 4. *Completeness of* **C**.
For every $\Gamma \cup \{A\} \subseteq \mathcal{W}^+$, if $\Gamma \vDash_{\mathbf{C}} A$ then $\Gamma \vdash_{\mathbf{C}} A$

Other important but easily provable properties of **C** are listed in the following theorem.

Theorem 5. **C** *is transitive, monotonic and reflexive.* **C***-proofs are recursive whence $Cn_{\mathbf{C}}$ is semi-recursive.*

Finally, observe that **C** is paraconsistent, in view of the fact that every premise set has a **C**-model; the model that makes all primitive formulas that occur in the premises or of which the negation occurs in the premises true and all other primitive formulas false. This model will satisfy the premises and falsify at least one formula (on the condition that not all primitive formulas or their negations occur in the premises—this condition is always true for premise sets $\{A, \neg A\}$).

Theorem 6. **C** *is paraconsistent, i.e. for every $A \in \mathcal{W}$, there is a $B \in \mathcal{W}^+$ such that $A, \neg A \nvdash_{\mathbf{C}} B$ and also $+A, +\neg A \nvdash_{\mathbf{C}} B$.*

6 A Strong Paraconsistent Version of CL

We now return to the relation between **C** and **CL**. We start by defining an extension of *consistent* **CL** (this is the logic that allows all the normal **CL**-consequences of consistent premises and does not give any consequence for inconsistent premise sets).

Definition 8. *The logic* $\mathbf{CL}^-$.
$\mathbf{CL}^-$ *uses the language $\mathcal{L}$ and where $\Gamma \cup \{A\} \in \mathcal{W}$, $\Gamma \vdash_{\mathbf{CL}^-} A$ iff $\Gamma \vdash_{\mathbf{C}} +A$*

The propositional fragment of this logic is neatly characterized in [1] as the logic **Q**. Proving that the propositional fragment of $\mathbf{CL}^-$ is indeed equivalent to this logic **Q** is quite straight forward but rather tedious.

I list the most important properties of $\mathbf{CL}^-$ (most of them are easily provable, except maybe the last one—for this property observe that $A, \neg A, (A \vee B) \vee \neg(A \vee B) \vdash_{\mathbf{C}} B$, which is provable by means of a proof of which the essential lines are $[\![A \vee B, \neg(A \vee B)]\!]$, $[\![A, B, \neg(A \vee B)]\!]$, and $[\![A, B, \neg A]\!]$).

Theorem 7. *1.* $\mathbf{CL}^-$ *is paraconsistent.*

2. $\mathbf{CL}^-$ *is not transitive.*
3. $\mathbf{CL}^-$ *is monotonic and reflexive.*
4. $\mathbf{CL}^-$*-proofs are recursive whence* $\vdash_{\mathbf{CL}^-}$ *is a semi-recursive relation (if the set of premises is recursive).*
5. $Cn_{\mathbf{CL}^-}(\Gamma \cup \{\neg A \vee A | A \in \mathcal{W}\}) = Cn_{\mathbf{CL}}(\Gamma)$.

The semi-recursiveness of $\vdash_{\mathbf{CL}^-}$ is quite important and remarkable. As far as my knowledge goes, there are no other existing paraconsistent logics that allow all the **CL**-consequences of consistent premise sets AND have a positive test whenever the set of premises is recursive. Most paraconsistent logics are weaker than consistent **CL** (they invalidate **CL**-rules like Disjunctive Syllogism or Addition). Those that are not weaker than consistent **CL** are usually non-monotonic (e.g. inconsistency adaptive logics, see [2], and consistent **CL** itself), which means that some consequences are revoked when the premise set is enriched with inconsistencies. Of course, this requires a full overview over the premises before one can be certain that some formula is a consequence. Consequently, these logics do not have a positive test (for infinite premise sets) whence they are not semi-recursive.

In what follows (and in the proofs in the appendix) let $\exists A$, where $A \in \mathcal{F}$, denote the existential closure of A and let $\Gamma \vDash_{\mathbf{L}} A \sqcup B$ denote that $M \models A$ or $M \models B$, for all **L**-models M such that $M \models \Gamma$

We now jump to the adequateness of $\mathbf{CL}^-$ (and hence also of **C**) with respect to **CL**. Proving soundness is straight forward: if every line $[\![\Delta]\!]$ is transformed into $\bigvee \Delta'$, where Δ' is Δ without $+$-symbols, every transformation is evidently a correct **CL**-consequence of the premises without $+$-symbols.

Theorem 8. *If* $\Gamma \vdash_{\mathbf{CL}^-} A$ *then* $\Gamma \vdash_{\mathbf{CL}} A$.

Completeness however is less straight forward. We first need two lemma's. The first lemma is easily provable in view of Theorem 8 and the fact that $A, \neg A \vdash_{\mathbf{CL}} B$.

Lemma 4. *If* $\Gamma \vdash_{\mathbf{C}} +\exists(D \wedge \neg D)$ *for some* $D \in \mathcal{F}$*, then* $Cn_{\mathbf{CL}}(\Gamma) = \mathcal{W}$.

Lemma 5. *If* $\Gamma \vDash_{\mathbf{C}} +A \sqcup +B \sqcup +C$*, then* $\Gamma \vDash_{\mathbf{C}} +A$*,* $\Gamma \vDash_{\mathbf{C}} +B$*,* $\Gamma \vDash_{\mathbf{C}} +C$ *or* $\Gamma, + * A, + * B \vDash_{\mathbf{C}} C$.

Now we finally have all the means to prove the (consistent premises) completeness of **C** and $\mathbf{CL}^-$ with respect to **CL**. Remark that this theorem is exactly the same as Theorem 2. The proof of this theorem is in the appendix.

Theorem 9. *If there is some* C *such that* $\Gamma \nvdash_{\mathbf{CL}} C$*, then* $\Gamma \vdash_{\mathbf{CL}^-} A$ *iff* $\Gamma \vdash_{\mathbf{CL}} A$.

7 A Relevant Logic Interpretation for the Lines of C-Proofs

The lines of **C**-proofs are easily interpretable as suggested by the soundness theorem: a line $[\![\Delta]\!]$ in a proof from premises Γ is interpreted as the expression "for all $A \in \Delta$, $\Gamma \cup \{\ddagger B \mid B \in \Delta - \{A\}\} \vdash_{\mathbf{C}} A$". This interpretation holds for all lines, but the rules of the proofs are not complete with respect to this interpretation, i.e. it is not the case that "if for every $A \in \Delta$ holds $\Gamma \cup \{\ddagger B \mid B \in \Delta - \{A\}\} \vdash_{\mathbf{C}} A$, then $\Gamma \vdash [\![\Delta]\!]$. A simple counterexample for this is the fact that $A \vee B, C \vee D \nvdash_{\mathbf{C}} [\![A, B, C, D]\!]$, while obviously $\{A \vee B, C \vee D, +\neg A, +\neg B, +\neg C\} \vdash_{\mathbf{C}} D$, $\{A \vee B, C \vee D, +\neg A, +\neg B, +\neg D\} \vdash_{\mathbf{C}} C$, $\{A \vee B, C \vee D, +\neg A, +\neg C, +\neg D\} \vdash_{\mathbf{C}} B$, and $\{A \vee B, C \vee D, +\neg B, +\neg C, +\neg D\} \vdash_{\mathbf{C}} A$.

Actually, these lines of **C**-proof are in fact relevant contraposable implications in disguise, where a line $[\![A, B, C, D]\!]$, for example, would, in a usual relevant logic like the logic **R**, be written as $*A \rightarrow (*B \rightarrow (*C \rightarrow D))$. The reason why the simple interpretation explained above is not adequate is precisely this relevant character of the lines. In particular, the problem is related to the fact that our proofs do not allow for weakening. This property/rule is usually expressed as $A \rightarrow (B \rightarrow A)$ and is not valid in relevant logics (if A is true anyhow, (the truth of) B is irrelevant for the truth of A). In the case of **C**-proofs weakening comes to deriving $[\![C, D]\!]$ from $[\![C]\!]$, which is evidently impossible. Another relevantly invalid property is $A \rightarrow (B \rightarrow (\neg A \rightarrow B))$. For our lines this comes to deriving $[\![A, B]\!]$ from lines $[\![A]\!]$ and $[\![B]\!]$, which is also not allowed. On the other hand all relevantly unproblematic consequences are allowed in **C**-proofs (modus ponens, transitivity, contraction, contraposition).

The following interpretation for lines of **C**-proofs is adequate with respect to **C**-proofs.

Definition 9. *Where $\Gamma, \Delta \subset \mathcal{W}^+$, $\Gamma \vDash_{\mathbf{C}} [\![\Delta]\!]$ iff $\langle \Gamma, \Delta \rangle \in X$, where X is the smallest set that satisfies the two following criteria*

1. $\langle \Gamma, \{A\} \rangle \in X$ *if* $\Gamma \vDash_{\mathbf{C}} A$,
2. $\langle \Gamma \cup \{A\}, \Delta \rangle \in X$ *if* $\langle \Gamma, \Delta \rangle \in X$,
3. $\langle \Gamma \cup \Gamma', \Delta \rangle$ *if* $\Gamma \vDash_{\mathbf{C}} A$ *and* $\langle \Gamma' \cup \{A\}, [\![\Delta]\!] \rangle$, *and*
4. $\langle \Gamma, \Delta \rangle$ *if, for every* $\Delta' \subset \Delta$, $\langle \Gamma \cup \{\ddagger B | B \in \Delta'\}, \Delta - \Delta' \rangle \in X$ *and, for every* $\Theta \subset \{\ddagger B | B \in \Delta'\}$, $\langle \Gamma \cup \Theta, \Delta - \Delta' \rangle \notin X$.

Theorem 10. *For every $\Gamma \cup \Delta \subseteq \mathcal{W}^+$, $\Gamma \vdash_{\mathbf{C}} [\![\Delta]\!]$ iff $\Gamma \vDash_{\mathbf{C}} [\![\Delta]\!]$.*

Although this interpretation is not deterministic, it is quite interesting as it seems to be an elegant means to define the semantics of quite a rich relevant implication without reference to worlds and accessibility relations.

I shall now make the relation between **C** and relevant conditionals precise. Let $\mathcal{W}^{\rightarrow}$ be the set of formulas defined by $\mathcal{W} \subset \mathcal{W}^{\rightarrow}$ and $A \rightarrow B \in \mathcal{W}^{\rightarrow}$ iff $A, B \in \mathcal{W}^{\rightarrow}$. We need to define a translation function tr before we can proceed to defining relevant theoremhood based on the lines of **C**-proofs.

1. where $n > 0$, $B \in \mathcal{W}$ and $A_1, \ldots A_n \in \mathcal{W}^{\rightarrow}$,
$$\mathrm{tr}(A_1 \rightarrow (A_2 \rightarrow (A_3 \rightarrow \ldots (A_n \rightarrow B) \ldots))) = [\![\ddagger\mathrm{tr}'(A_1), \ddagger\mathrm{tr}'(A_2), \ddagger\mathrm{tr}'(A_3), \ldots, \ddagger\mathrm{tr}'(A_n), +B]\!]$$
2. where $A \in \mathcal{W}$, $\mathrm{tr}(A) = +A$ and $\mathrm{tr}'(A) = A$
3. where $A_1, A_2 \in \mathcal{W}^{\rightarrow}$ $\mathrm{tr}'(A_1 \rightarrow A_2) = \neg\mathrm{tr}'(A_1) \vee \mathrm{tr}'(A_2)$

Definition 10. *Let the logic* $\vdash_{\mathbf{Cr}}$ *be defined by:*

- *the set of formulas of* **Cr** *is* $\mathcal{W}^{\to}$,
- *where* $A \in \mathcal{W}^{\to}$, $\vdash_{\mathbf{Cr}} A$ *iff* $\vdash_{\mathbf{C}} \mathrm{tr}(A)$, *and*
- *only theoremhood of* **Cr** *is defined, no consequence relation.*

It is easy to check that: (in the right column the corresponding translation to a **C**-theorem is mentioned)

$\nvdash_{\mathbf{Cr}} B \to (A \to B)$	$\nvdash_{\mathbf{C}} \llbracket + * B, + * A, +B \rrbracket$
$\nvdash_{\mathbf{Cr}} A \to (B \to B)$	$\nvdash_{\mathbf{C}} \llbracket + * A, + * B, +B \rrbracket$
$\nvdash_{\mathbf{Cr}} A \to (\neg B \vee B)$	$\nvdash_{\mathbf{C}} \llbracket + * A, +(\neg B \vee B) \rrbracket$
$\nvdash_{\mathbf{Cr}} B \to (\neg B \to A)$	$\nvdash_{\mathbf{C}} \llbracket + * B, +B, +A \rrbracket$
$\nvdash_{\mathbf{Cr}} B \to (A \to (\neg B \to A)$	$\nvdash_{\mathbf{C}} \llbracket + * B, + * A, +B, +A \rrbracket$
$\vdash_{\mathbf{Cr}} A \to A$	$\vdash_{\mathbf{C}} \llbracket + * A, +A \rrbracket$
$\vdash_{\mathbf{Cr}} A \to (A \vee B)$	$\vdash_{\mathbf{C}} \llbracket + * A, +(A \vee B) \rrbracket$
$\vdash_{\mathbf{Cr}} (A \wedge B) \to A$	$\vdash_{\mathbf{C}} \llbracket + * (A \wedge B), +A \rrbracket$
$\vdash_{\mathbf{Cr}} A \to (B \to (A \wedge B))$	$\vdash_{\mathbf{C}} \llbracket + * A, + * B, +(A \wedge B) \rrbracket$
where $A \in \mathcal{W}$ is a **CL**-theorem, $\vdash_{\mathbf{Cr}} A$	$\vdash_{\mathbf{C}} \llbracket +A \rrbracket$
$\vdash_{\mathbf{Cr}} (A \to B) \to ((B \to C) \to (A \to C))$	$\vdash_{\mathbf{C}} \llbracket + * (*A \vee B), + * (*B \vee C), + * A, +C \rrbracket$
$\vdash_{\mathbf{Cr}} (A \to (B \to C)) \to (B \to (A \to C))$	$\vdash_{\mathbf{C}} \llbracket + * (*A \vee (*B \vee C)), + * B, + * A, +C \rrbracket$

It is still unclear whether **Cr** is a fully relevant logic, but the above properties are quite promising. All straight forward irrelevances of **CL** (paradoxes of material implication) are eliminated for the →-implication, while at the same time it is crystal clear that → is a fairly strong conditional (all axioms of the standard relevant logic **R** are valid). Although it seems likely that **Cr** is a useful relevant logic, it has an elegant proof theory which has an adequate semantics that does not refer to worlds or accessibility relations. This is a rather remarkable result.

Remark that this conditional has some unusual properties. For example, Disjunctive Syllogism is valid (i.e. $\vdash_{\mathbf{Cr}} (A \vee B) \to (\neg A \to B)$), and so are Addition (i.e. $\vdash_{\mathbf{Cr}} A \to (A \vee B)$), Transitivity (i.e. $\vdash_{\mathbf{Cr}} (A \to B) \to ((B \to C) \to (A \to C)))$, and Modus Ponens (i.e. $\vdash_{\mathbf{Cr}} A \to ((A \to B) \to B)$). From the validity of these rules, one would expect that Ex Falso Quodlibet (i.e. $A \to (\neg A \to B)$) is also validated in **Cr**, but this is NOT the case. This is possible due to the fact that the metatheoretic version of Modus Ponens (from $\vdash A$ and $\vdash A \to B$, derive $\vdash B$) is NOT valid in **Cr**.

8 Procedures for Proof Generation

In the first section, I mentioned that the ideas for the logic **C** are inspired by the goal directed proof system defined by Batens and Provijn. The main objective of those goal directed proofs is to push the heuristics a reasoner uses to construct a proof for a given formula into the proof itself. The goal directed proofs do not allow useless (but essentially correct) proof steps. For example, if one wants to derive q from p and $p \supset q$, deriving $p \vee r$ will (although evidently correct in **CL**) never be considered as a wise step in a proof. Goal directed proofs give a formal criterion to distinguish between potentially useful and useless steps in a proof with a certain goal. Evidently, these goal directed proofs, in which one avoids useless steps, can easily be turned into algorithms that construct proofs. Hence, they also form (partial) decision methods for logical consequence.

My aim was precisely to eliminate the goal dependency from the goal directed proofs (and thereby making transitive reasoning and a usual semantics and axiomatization for $\mathbf{CL}^{-}$ possible). So I had to lose part of the advantage and the original aim of the goal directed proofs. However, the goal directed element can easily be added on top of the **C1**-proofs. Making **C1**-proofs as elegant and insightful as the original goal directed proofs is just a matter of adding some minor heuristic information to the proofs; the logical rules do not need to change. Such an enterprise, however, is not substantial for my present purpose.

References

[1] Batens, D.: It might have been classical logic. Logique et Analyse, `http://logica.ugent.be/centrum/preprints/PCLs.pdf`

[2] Batens, D.: Inconsistency-adaptive logics. In: Orłowska, E. (ed.) Logic at Work. Essays Dedicated to the Memory of Helena Rasiowa, pp. 445–472. Physica Verlag (Springer) (1999)

[3] Batens, D.: A formal approach to problem solving. In: Delrieux, C., Legris, J. (eds.) Computer Modeling of Scientific Reasoning, pp. 15–26. Universidad Nacional del Sur, Bahia Blanca (2003)

[4] Batens, D., Provijn, D.: Pushing the search paths in the proofs. A study in proof heuristics. Logique et Analyse 173-175, 113–134 (2001) (appeared 2003)

[5] Besnard, P., Hunter, A.: Quasi-classical logic: Non-trivializable classical reasoning from incosistent information. In: Froidevaux, C., Kohlas, J. (eds.) ECSQARU 1995. LNCS, vol. 946, pp. 44–51. Springer, Heidelberg (1995)

[6] Craig, W.: Linear reasoning. A new form of the Herbrand-Gentzen theorem. The Journal of Symbolic Logic 22(3), 250–268 (1957)

[7] D'Agostino, M., Floridi, L.: The enduring scandal of deduction. Synthese 167(2) (2009)

[8] Gentzen, G.: Untersuchungen über das logische schließen. ii. Mathematische Zeitschrift 39, 405–431 (1935) ISSN 0025-5874

[9] Hunter, A.: Reasoning with contradictory information using quasi-classical logic. Journal of Logic and Computation 10, 677–703 (1999)

[10] Hunter, A.: A semantic tableau version of first-order quasi-classical logic. In: Benferhat, S., Besnard, P. (eds.) ECSQARU 2001. LNCS (LNAI), vol. 2143, pp. 544–555. Springer, Heidelberg (2001)

[11] Jago, M.: Logical information and epistemic space. Synthese 167(2) (2009)

[12] Provijn, D.: Prospectieve dynamiek. Filosofische en technische onderbouwing van doelgerichte bewijzen en bewijsheuristieken. PhD thesis, Universiteit Gent, Belgium (2005) (unpublished PhD thesis)

Appendix: Metaproofs

Sketch of the Proof of Lemma 2

Suppose $\Delta' \subset \Delta$, $\Gamma \cup \{\ddagger B | B \in \Delta'\} \vdash_{\mathbf{C}} \llbracket \Delta - \Delta' \rrbracket$ and for all $\Theta \subset \{\ddagger B | B \in \Delta'\}$, $\Gamma \cup \Theta \nvdash_{\mathbf{C}} \llbracket \Delta - \Delta' \rrbracket$. Hence there is a proof in which $\llbracket \Delta - \Delta' \rrbracket$ is derived and every $\llbracket \ddagger B \rrbracket$, where $B \in \Delta'$, is used as a local premise in the derivation tree of $\llbracket \Delta - \Delta' \rrbracket$. Now it is possible to replace every such $\llbracket \ddagger B \rrbracket$ by $\llbracket \ddagger B, B \rrbracket$ (remember that this is an axiom). The rest of the proof can remain the same, apart from the fact that for every line which is in the derivation tree on a branch that contains a line $\llbracket \ddagger B \rrbracket$, where $B \in \Delta'$, B should be added as an element of that line. The result will be that the line $\llbracket \Delta - \Delta' \rrbracket$ is transformed into $\llbracket \Delta \rrbracket$ whence the proof for $\Gamma \cup \{\ddagger B | B \in \Delta'\} \vdash_{\mathbf{C}} \llbracket \Delta - \Delta' \rrbracket$ is transformed into a proof for $\Gamma \vdash_{\mathbf{C}} \llbracket \Delta \rrbracket$.

Sketch of the Proof of Theorem 4

Let $\mathcal{L}_p$ be exactly as $\mathcal{L}_o$ except for the replacement of $\mathcal{O}$ by $\mathcal{O}'$, which is an arbitrary, countably infinite subset of $\mathcal{O}$. $\mathcal{W}_p$, $\mathbb{P}_p$, $\mathcal{L}_p^+$, $\mathcal{W}_p^+$, and $\mathbb{P}_p^+$ are defined analogously. Let $\mathcal{D}_p =_{\text{df}} \{\llbracket \Delta \rrbracket \, | \Delta \subset \mathcal{W}_p^+\}$.

Let a $\mathbf{C}*$-proof from $\Gamma \in \mathcal{W}_p^+ \cup \mathcal{D}_p$ be a list of lines such that every line is either a line in a $\mathbf{C}$-proof from $\Gamma \cap \mathcal{W}_p^+$, an element of $\Gamma \cap \mathcal{D}_p$ or a result of applying the rules TRANS, or UG to preceding lines of the proof. Where $\Gamma \cup \{A\} \in \mathcal{W}_p^+ \cup \mathcal{D}_o$, define $\vdash_{\mathbf{C}*}$ by $\Gamma \vdash_{\mathbf{C}*} A$ iff there is a $\mathbf{C}*$-proof for A from Γ. Obviously, with this definition, for every $\Gamma \cup \{A\} \subseteq \mathcal{W}_p$, $\Gamma \vdash_{\mathbf{C}} A$ iff $\Gamma \vdash_{\mathbf{C}*} A$, but remark that e.g. $\llbracket Pa \rrbracket \vdash_{\mathbf{C}*} +\neg\exists x Px$, whereas evidently $Pa \nvdash_{\mathbf{C}} +\neg\exists x \neg Px$ and $Pa \nvdash_{\mathbf{C}*} +\neg\exists x \neg Px$.

Suppose $\Gamma \nvdash_{\mathbf{C}} A$, where $\Gamma \cup A \subseteq \mathcal{W}^+$, whence $\Gamma \nvdash_{\mathbf{C}*} A$. Let $\langle B_1, B_2, \ldots \rangle$ be an enumeration of all elements of $\mathcal{W}_p^+ \cup \{\llbracket A, C \rrbracket \mid C \in \mathcal{W}_p^+\}$ such that if $B_i = \exists \beta C(\beta)$ then $B_{i+1} = C(\alpha)$ and if $B_i = \llbracket A, \exists \beta C(\beta) \rrbracket$ then $B_{i+1} = \llbracket A, C(\alpha) \rrbracket$, where $\alpha \in \mathcal{O}'$ does not occur in $\{B_1, \ldots, B_i\}$. Define:

$$\Delta_0 = Cn_{\mathbf{C}}(\Gamma)$$

$$\begin{cases} \Delta_{i+1} = Cn_{\mathbf{C}*}(\Delta_i \cup \{B_{i+1}\}) & \text{if } A \notin Cn_{\mathbf{C}*}(\Delta_i \cup \{\llbracket B_{i+1} \rrbracket\}) \\ \Delta_{i+1} = \Delta_i & \text{otherwise} \end{cases}$$

$$\Delta = \Delta_0 \cup \Delta_1 \cup \ldots$$

We show that Δ has the following properties:

1. $\Gamma \subseteq \Delta$. Immediately.
2. $A \notin \Delta$. Immediately.
3. Δ is deductively closed. Immediately.
4. Δ is ω-complete i.e. if $\exists \alpha A(\alpha) \in \Delta$ then $A(\beta) \in \Delta$ for at least one $\beta \in \mathcal{C} \cup \mathcal{O}$. Suppose $\exists \alpha A(\alpha) \in \Delta$, $B_i = \exists \alpha A(\alpha)$ and there is no $\beta \in \mathcal{C} \cup \mathcal{O}$ such that $A(\beta) \in \Delta$. But then also $\Delta_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}} A$, with $B_{i+1} = A(\beta)$ for some $\beta \in \mathcal{O}$ that does not occur in Δ_i. Hence, in view of Lemma 2, $\Delta_i \vdash_{\mathbf{C}} A$ or $\Delta_i \vdash_{\mathbf{C}} \llbracket + * A(\beta), A \rrbracket$. The former disjunct is impossible in view of property

2 and the latter disjunct entails $\Delta_i \vdash_{\mathbf{C}} [\![+\neg\exists\alpha A(\alpha), A]\!]$ (β does not occur in $\Delta_i \cup \{A\}$), whence $\Delta_i \vdash [\![A]\!]$ (by $\Delta_i \vdash_{\mathbf{C}} [\![\exists\alpha A(\alpha)]\!]$ and TRANS), which is also impossible in view of property 2.

5. Δ is $\omega*$-complete i.e. if $[\![A, \exists\alpha C(\alpha)]\!] \in \Delta$ then $[\![A, C(\beta)]\!] \in \Delta$ for at least one $\beta \in \mathcal{C} \cup \mathcal{O}$. Suppose $[\![A, \exists\alpha C(\alpha)]\!] \in \Delta$, $B_i = [\![\exists\alpha C(\alpha)]\!]$ and there is no $\beta \in \mathcal{C} \cup \mathcal{O}$ such that $C(\beta) \in \Delta$. But then also $\Delta_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}*} A$, with $B_{i+1} = [\![A, C(\beta)]\!]$ for some $\beta \in \mathcal{O}$ that does not occur in Δ_i. Hence, in view of $\Delta_i \nvdash_{\mathbf{C}*} A$, $\Delta_i \cup [\![C(\beta)]\!] \vdash_{\mathbf{C}*} A$ and therefore $\Delta_i \cup [\![\exists\alpha A(\alpha), A]\!] \vdash_{\mathbf{C}} A$ (consider a proof that contains a line j saying $[\![A, \exists\alpha C(\alpha)]\!]$. We can add a line $[\![C(\beta), +*C(\beta)]\!]$. Because $\Delta_i \cup [\![C(\beta)]\!] \vdash_{\mathbf{C}*} A$, we can add a number of lines resulting in $[\![A, +*C(\beta)]\!]$ and therefore also $[\![A, +\neg\exists\alpha C(\alpha)]\!]$ can be added, which with line j allows us to conclude $[\![A]\!]$.) This would entail $\Delta \cup [\![\exists\alpha A(\alpha), A]\!] \vdash_{\mathbf{C}*} A$ and hence also $\Delta \vdash_{\mathbf{C}*} A$, which is impossible in view of property 2.
6. For every $C \in \mathcal{W}_o^+$ holds that if $C \notin \Delta$, then $\ddagger C \in \Delta$. Suppose $C \notin \Delta$ and $\ddagger C \notin \Delta$. Then there are $i, j \in \mathbb{N}$ such that $\Delta_i \cup \{C\} \vdash_{\mathbf{C}} A$ and $\Delta_j \cup \{\ddagger C\} \vdash_{\mathbf{C}} A$. But then, by Corollary 3, also $\Delta_i \cup \Delta_j \vdash_{\mathbf{C}} A$, which is impossible in view of property 2.

Where $\alpha \in \mathcal{C} \cup \mathcal{O}'$, let $\overline{\alpha} = \{\alpha\} \cup \{\beta \mid \alpha = \beta \in \Delta\}$. Observe that the set $\{\overline{\alpha} | \alpha \in \mathcal{C} \cup \mathcal{O}'\}$ is a partition of the set $\mathcal{C} \cup \mathcal{O}'$.

Define M as the triple $\langle v, w, D\rangle$, where $D = \{\overline{\alpha} \mid \alpha \in \mathcal{C} \cup \mathcal{O}\}$, v is defined by:

1. for every $\alpha \in \mathcal{C} \cup \mathcal{O}$: $v(\alpha) = \overline{\alpha}$,
2. $v(\sigma) = 1$ iff $\sigma \in \Delta$,
3. $v(\sigma_\neg) = 1$ iff $\neg\sigma \in \Delta$,
4. for every $\pi^r \in \mathcal{P}^r$: $v(\pi^r) = \{\langle v(\alpha_1), \ldots, v(\alpha_r)\rangle \mid \pi\alpha_1 \ldots \alpha_r \in \Delta\}$,
5. for every $\pi^r \in \mathcal{P}^r$: $v(\pi^r_\neg) = \{\langle v(\alpha_1), \ldots, v(\alpha_r)\rangle \mid \neg\pi\alpha_1 \ldots \alpha_r \in \Delta\}$,
6. $v(\neg \cdot = \cdot) = \{\langle v(\alpha), v(\beta)\rangle \mid \neg\alpha = \beta \in \Delta\}$,
7. $v(+\neg \cdot = \cdot) = \{\langle v(\alpha), v(\beta)\rangle \mid +\neg\alpha = \beta \in \Delta\}$, and
8. $v(\cdot = \cdot) = \{\langle v(\alpha), v(\beta)\rangle \mid \alpha = \beta \in \Delta\}$

and $w : \mathcal{W}_o^+ \mapsto \{0, 1\}$ is defined by

1. for every $B \in \mathcal{W}_o$: $w(B) = 1$ iff $B \in \Delta$, and
2. for every $B \in \mathcal{W}_o$: $w(+B) = 1$ iff $+B \in \Delta$.

Note that, for all $B \in \mathbb{P}_o^\neg \cup \{A \mid \alpha, \beta \in \mathcal{C} \cup \mathcal{O}; A = \alpha = \beta \text{ or } A = +\neg\alpha = \beta \text{ or } A = +\neg\alpha = \beta\}$,

$$I_v(B) \text{ iff } B \in \Delta\,. \tag{1}$$

For all $B \in \mathcal{W}_o$, if $v_M(B) = 1$ then $(w(B) = 1$ or $I_v(B))$ and therefore $B \in \Delta$. We obtain,

$$\text{if } B \in \mathcal{W}_o \text{ and } v_M(B) = 1 \text{ then } B \in \Delta\,. \tag{2}$$

For all $B \in \mathcal{W}_o$, if $B \in \Delta$ then $w(B) = 1$ and, when B is of the form $\neg\alpha = \beta$, $I_v(B)$. Consequently, $B \in \Delta$. We obtain,

$$\text{if } B \in \mathcal{W}_o \text{ and } +B \in \Delta \text{ then } v_M(+B) = 1\,. \tag{3}$$

We shall prove that, for all $B \in \mathcal{W}_o^+$,

$$v_M(B) = 1 \text{ iff } B \in \varDelta\,. \tag{4}$$

by means of an induction on the complexity of B (the complexity of B is computed by counting all occurrences of $\neg\neg$, $\vee$ and $\exists$). Note that I omit all statements that immediately follow from (2) or (3).

For the induction basis, we have the following cases.

- $B \in \mathbb{P}$. If $B \in \varDelta$ then $w(B) = 1$ (5) and by (1), $I_v(B)$ (6). (5) and (6) together entail $v_M(B) = 1$.
- $B = \neg C$ and $C \in \mathbb{P}$. If $\neg C \in \varDelta$ then $w(\neg C)$ (7) and, by (1), $I_v(\neg C)$ (8). (7) and (8) together entail $v_M(\neg C) = 1$.
- $B = \alpha = \beta$. If $\alpha = \beta \in \varDelta$ then $v(\alpha) = \overline{\alpha} = \overline{\beta} = v(\beta)$ (9). If $\alpha = \beta \in \varDelta$ then, by (1), $I_v(\alpha = \beta)$ (10). (9) and (10) together entail $v_M(\alpha = \beta) = 1$.
- $B = \neg\alpha = \beta$. If $\neg\alpha = \beta \in \varDelta$ then $w(\neg\alpha = \beta) = 1$ (11) and, by (1), $I_v(\neg\alpha = \beta)$ (12). If $v(\alpha) = v(\beta)$ then, by (12), $I_v(\neg\alpha = \alpha)$ whence $T_M(v(\alpha))$ holds. (11) and ($v(\alpha) \neq v(\beta)$ or $T_M(v(\alpha))$) together entail $v_M(\neg\alpha = \beta) = 1$.
- $B = +C$ and $C \in \mathbb{P}$. If $\ddagger[v_M(\neg C) = 1]$ then $w(\neg C) = 0$ or $w(+C) = 1$ or $I_v(C)$. If $w(\neg C) = 0$ then $\neg C \notin \varDelta$ whence, by property 6, $+C \in \varDelta$. If $w(+C) = 1$ then $+C \in \varDelta$. If $I_v(C)$ then, by (1), $C \in \varDelta$ whence, by $C \vdash_{\mathbf{C}} +C$, $+C \in \varDelta$.
- $B = +\neg C$ and $C \in \mathbb{P}$. If $\ddagger[v_M(C) = 1]$ then $w(C) = 0$ or $w(+\neg C) = 1$ or not $I_v(C)$ or $I_v(\neg C)$. If $w(C) = 0$ then $C \notin \varDelta$ whence, by property 6, $+\neg C \in \varDelta$. If $w(+\neg C) = 1$ then $+\neg C \in \varDelta$. If $I_v(\neg C)$ then, by (1), $\neg C \in \varDelta$ and, by $\neg C \vdash_{\mathbf{C}} +\neg C$, also $+\neg C \in \varDelta$. Finally, if not $I_v(C)$ then, by (1), $C \notin \varDelta$ whence, by property 6, $+\neg C \in \varDelta$.
- $B = +\alpha = \beta$. If $\ddagger[v_M(\neg\alpha = \beta) = 1]$ then $w(\neg\alpha = \beta) = 0$ or $w(+\alpha = \beta) = 1$ or $v(\alpha) = v(\beta)$. If $w(\neg\alpha = \beta) = 0$ then $\neg\alpha = \beta \notin \varDelta$ whence, by property 6, $+\alpha = \beta \in \varDelta$. If $w(+\alpha = \beta) = 1$ then $+\alpha = \beta \in \varDelta$. Finally, if $v(\alpha) = v(\beta)$ then also $\alpha = \beta \in \varDelta$ or $\alpha = \beta$. In both cases $+\alpha = \beta \in \varDelta$.
- $B = +\neg\alpha = \beta$. If $\ddagger[v_M(\alpha = \beta) = 1]$ then not $I_v(\alpha = \beta)$ or $I_v(+\neg\alpha = \beta)$ or $v(\alpha) \neq v(\beta)$ or $T_M(v(\alpha))$. If not $I_v(\alpha = \beta)$ then $\alpha = \beta \notin \varDelta$ whence, by property 6, $+\neg\alpha = \beta \in \varDelta$. If $I_v(+\neg\alpha = \beta)$ then, by (1) $+\alpha = \beta \in \varDelta$. Either $v(\alpha) \neq v(\beta)$ or $v(\alpha) = v(\beta)$. If $v(\alpha) \neq v(\beta)$ then $\alpha = \beta \notin \varDelta$ whence, by property 6, $+\neg\alpha = \beta \in \varDelta$. Suppose now $v(\alpha) = v(\beta)$ and $T_M(v(\alpha))$. Then either $I_v(\neg\alpha = \alpha)$ and therefore $I_v(\neg\alpha = \beta)$ whence, by 1, $\neg\alpha = \beta \in \varDelta$ and thus $+\neg\alpha = \beta \in \varDelta$, or there is an r-ary predicate π and $\alpha_1, \ldots, \alpha_{r-1} \in D$, such that $\langle \alpha_1, \ldots, \alpha_i, v(\alpha), \alpha_{i+1}, \ldots, \alpha_{r-1} \rangle \in v(\pi) \cap v(\pi_\neg))$'. Hence there is a $C(\alpha) \in \mathbb{P}_o$ such that $I_v(C(\alpha))$ and $I_v(\neg C(\alpha))$, and therefore, by $v(\alpha) = v(\beta)$, also $I_v(\neg C(\beta))$ and, by (1), $\{C(\alpha), \neg C(\beta)\} \subset \varDelta$. This entails $+\neg\alpha = \beta \in \varDelta$ (AS7).

For the **induction step**, suppose $v_M(C) = 1$ iff $C \in \varDelta$, for every C of complexity lower than c. We prove that $v_M(B) = 1$ iff $B \in \varDelta$ for every B of complexity c.

Given (2) and (3), we only need to prove that if $B \in \varDelta$ then $v_M(B) = 1$ where $B \in \mathcal{W}_o$ and if $\ddagger[v_M(\ddagger B) = 1]$ then $B \in \varDelta$ where $B = +C$ and $C \in \mathcal{W}_o$.

- $B = D \vee E$. Immediately in view of $D \vee E, +*D \vdash_{\mathbf{C}} E$ and $D \vee E, +*E \vdash_{\mathbf{C}} D$.
- $B = \neg(D \vee E)$. Immediately in view of $\neg(D \vee E) \vdash_{\mathbf{C}} *D$ and $\neg(D \vee E) \vdash_{\mathbf{C}} *E$.
- $B = \exists \alpha D(\alpha)$. Immediately in view of the ω-completeness of Δ.
- $B = \neg\exists \alpha D(\alpha)$. Immediately in view of $\neg\exists \alpha D(\alpha) \vdash_{\mathbf{C}} *D(\beta)$ for all $\beta \in \mathcal{C} \cup \mathcal{O}$.
- $B = +(D \vee E)$. Immediately in view of $+D \vdash_{\mathbf{C}} +(D \vee E)$ and $+E \vdash_{\mathbf{C}} +(D \vee E)$.
- $B = +\neg(D \vee E)$. Immediately in view of $+D, +E \vdash_{\mathbf{C}} +(D \wedge E)$.
- $B = \neg\neg D$. Immediately in view of $\neg\neg D \vdash_{\mathbf{C}} D$.
- $B = +\neg\neg D$. Immediately in view of $+D \vdash_{\mathbf{C}} +\neg\neg D$.
- $B = +\exists \alpha D(\alpha)$. Immediately in view of $D(\beta) \vdash_{\mathbf{C}} \exists \alpha D(\alpha)$ for all $\beta \in \mathcal{C} \cup \mathcal{O}$.
- $B = +\neg\exists \alpha D(\alpha)$. Suppose $\ddagger[v_M(\ddagger B) = 1]$ and thus $w(\exists \alpha D(\alpha)) = 0$ or $w(+\neg\exists \alpha D(\alpha)) = 1$ or for all $\beta \in \mathcal{C} \cup \mathcal{O}$, $v_M(+ * D(\beta)) = 1$. The two first cases are obvious. In the last case, by the induction hypothesis, $+*D(\beta) \in \Delta$ for all $\beta \in \mathcal{C} \cup \mathcal{O}$ (13). Suppose $+\neg\exists \alpha D(\alpha) \notin \Delta$. Hence there is an i such that $\Delta_i \cup \{+\neg\exists \alpha D(\alpha)\} \vdash_{\mathbf{C}} A$ whence, in view of Lemma 2, $\Delta_i \vdash_{\mathbf{C}} [\![A, \exists \alpha D(\alpha)]\!]$ and therefore $[\![A, \exists \alpha D(\alpha)]\!] \in \Delta$. In view of the $\omega*$-completeness of Δ there exists a γ such that $[\![A, D(\gamma)]\!] \in \Delta$. As a consequence there is a, $\Delta_i \cup \{+ * D(\gamma)\} \vdash_{\mathbf{C}} A$ whence $+ * D(\gamma) \notin \Delta$, which contradicts (13).

By (4) and properties 1 and 2, we obtain $\Gamma \nvdash_{\mathbf{C}} A$.

Sketch of the Proof of Theorem 9

Suppose

$$\Gamma \nvdash_{\mathbf{C}} +A \tag{14}$$

where $\Gamma \cup +A \subseteq \mathcal{W}^+$ and

$$\text{there is some } C \text{ such that } \Gamma \nvdash_{\mathbf{CL}} C\,. \tag{15}$$

The latter supposition entails

$$\Gamma \nvdash_{\mathbf{C}} +\exists(D \wedge \neg D) \text{ for every } D \in \mathcal{W}_p\,, \tag{16}$$

in view of Lemma 4.

Let $L = \langle B_1, B_2, \ldots \rangle$ be an infinite list of all members of $\mathcal{W}_p$ (for the definition of $\mathcal{W}_p$, see the previous proof) such that if $B_i = \exists \beta C(\beta)$ then $B_{i+1} = C(\alpha)$, where $\alpha \in \mathcal{O}'$ does not ocuur in Δ_i.

$$\Delta_1 = Cn_{\mathbf{C}}(\Gamma) \tag{17}$$

$$\Delta_{i+1} = \begin{cases} Cn_{\mathbf{C}}(\Delta_i \cup \{B_{i+1}\}) & \text{if } \Delta_i \cup \{B_{i+1}\} \nvdash_{\mathbf{C}} +A \text{ and there is} \\ & \text{no } C \in \mathcal{W}_p \text{ such that} \\ & \Delta_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}} +\exists(C \wedge \neg C) \\ Cn_{\mathbf{C}}(\Delta_i \cup \{\ddagger B_{i+1}\}) & \text{otherwise} \end{cases} \tag{18}$$

$$\Delta = \Delta_1 \cup \Delta_2 \cup \ldots \tag{19}$$

We prove by means of mathematical induction that, for every $i > 1$,

$$+A \notin \mathit{\Delta}_i \text{ and there is no } D \in \mathcal{F} \text{ such that } +\exists(D \wedge \neg D) \in \mathit{\Delta}_i\,. \tag{20}$$

For the basic case, $+A \notin \mathit{\Delta}_1$ (from supposition (14)) and there is no $D \in \mathcal{W}_p$ such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_1$ (from (16)).

For the induction step we need to prove that if "$+A \notin \mathit{\Delta}_i$ and there is no $D \in \mathcal{W}_p$ such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_i$", then "$+A \notin \mathit{\Delta}_{i+1}$ and there is no $D \in \mathcal{W}_p$ such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_{i+1}$". Suppose that the antecedent holds. There are two cases.

Case 1: (i) $\mathit{\Delta}_i \cup \{B_{i+1}\} \nvdash_{\mathbf{C}} +A$ and (ii) there is no $C \in \mathcal{F}_p$ such that $\mathit{\Delta}_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}} +\exists(C \wedge \neg C)$. In this case $\mathit{\Delta}_{i+1} = Cn_{\mathbf{C}}(\mathit{\Delta}_i \cup \{B_{i+1}\})$ whence $+A \notin \mathit{\Delta}_{i+1}$ (from (i)) and there is no $D \in \mathcal{W}_p$ such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_{i+1}$ (from (ii)).

Case 2: $\mathit{\Delta}_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}} +A$ or there is a $C \in \mathcal{F}$ such that $\mathit{\Delta}_i \cup \{B_{i+1}\} \vdash_{\mathbf{C}} +\exists(C \wedge \neg C)$. In this case $\mathit{\Delta}_{i+1} = Cn_{\mathbf{C}}(\mathit{\Delta}_i \cup \{\ddagger B_{i+1}\})$ and there is a $C \in \mathcal{F}$ such that

$$\mathit{\Delta}_i \cup \{B_{i+1}\} \vDash_{\mathbf{C}} +A \sqcup +\exists(C \wedge \neg C)\,. \tag{21}$$

Suppose now $+A \in \mathit{\Delta}_{i+1}$ or there is a $D \in \mathcal{W}_p$ such that $+D \in \mathit{\Delta}_{i+1}$ and $+\neg D \in \mathit{\Delta}_{i+1}$. But then there would also be a $D \in \mathcal{W}_p$, such that (by $+A \vdash_{\mathbf{C}} +(A \vee \exists(D \wedge \neg D))$ and $+D, +\neg D \vdash_{\mathbf{C}} +(A \vee \exists(D \wedge \neg D))$)

$$\mathit{\Delta}_i \cup \{\ddagger B_{i+1}\} \vDash_{\mathbf{C}} +A \sqcup +\exists(D \wedge \neg D)\,. \tag{22}$$

From (21) and (22), Corollary 3 warrants that

$$\mathit{\Delta}_i \vDash_{\mathbf{C}} +A \sqcup +\exists(D \wedge \neg D) \sqcup +\exists(C \wedge \neg C)\,. \tag{23}$$

This entails (by Lemma 5)

$$\begin{gathered} \mathit{\Delta}_i \vDash_{\mathbf{C}} +A \text{ or } \mathit{\Delta}_i \vDash_{\mathbf{C}} +\exists(D \wedge \neg D) \text{ or} \\ \mathit{\Delta}_i \vDash_{\mathbf{C}} +\exists(C \wedge \neg C) \text{ or} \\ \mathit{\Delta}_i \cup \{+\neg\exists(C \wedge \neg C), +\neg\exists(D \wedge \neg D)\} \vDash_{\mathbf{C}} A\,. \end{gathered} \tag{24}$$

As $\vdash_{\mathbf{C}}$ is complete with respect to its semantics and as $+\neg\exists(E \wedge \neg E) \in \mathit{\Delta}_i$ for every $E \in \mathcal{F}$ (the reader can check that $\vDash_{\mathbf{C}} +\neg\exists(E \wedge \neg E) \in \mathit{\Delta}_i$), this is in contradiction with the induction hypothesis. Consequently, $+A \notin \mathit{\Delta}_{i+1}$ and there is no $D \in \mathcal{F}$ such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_{i+1}$.

Now we have

$$A \notin \mathit{\Delta}\,, \tag{25}$$

$$\text{there is no } D \in \mathcal{W}_p \text{ such that } +D, +\neg D \in \mathit{\Delta}\,, \tag{26}$$

(otherwise there would be an i such that $+\exists(D \wedge \neg D) \in \mathit{\Delta}_i$)) and

$$\text{for every } D \in \mathcal{W}_p,\ D \in \mathit{\Delta} \text{ or } \ddagger D \in \mathit{\Delta}\,. \tag{27}$$

(26) and (27) together entail

$$\text{for every } D \in \mathcal{W}_p, \text{ if } +D \in \Delta \text{ then } \ddagger D \in \Delta\,. \tag{28}$$

Consequently (remember that $\vdash_{\mathbf{C}} +(D \vee \neg D))$ $D \vee \neg D \in \Delta$, for every $D \in \Delta$, whence, by Theorem 7 item 5 and the fact that Δ is **C**-deductively closed,

$$Cn_{\mathbf{CL}}(\Delta \cap \mathcal{W}_p) = \Delta \cap \mathcal{W}_p\,. \tag{29}$$

Using (25-29) one can show by means of the usual methods that $\Delta \cap \mathcal{W}_p$ is a **CL**-model set (i.e. it is maximally non-trivial, **CL**-deductively closed and ω-complete). Hence there exists a **CL**-model such that, for all $A \in \mathcal{W}$, $M \models A$ iff $A \in \Delta$. In view of (25), we finally obtain

$$\Gamma \nvdash_{\mathbf{CL}} A\,.$$

How Much Expressive Power Is Needed for Natural Language Temporal Indexicality?*

Igor Yanovich

MIT
yanovich@mit.edu

Abstract. The paper studies how much expressive power beyond the capabilities of the simple Priorean temporal language $\mathbf{K}_t$ is needed to give proper translation to natural language examples by Kamp and Vlach which are extensively used in the linguistic and philosophical literature as forcing the use of quite expressive languages, all the way up to full two-sorted FOL. It turns out that when examined carefully, the examples in question only require a quite mild Kamp- and Cresswell-style system with **now** and **then** operators, or, equivalently, hybrid $\mathbf{K}_t + \downarrow + @$. The paper generalizes the earlier results showing that in the propositional case, **now** and **then** do not increase the power of $\mathbf{K}_t$. For the first-order case, a notion of FOL path bisimulation for first-order $\mathbf{K}_t^{FO}$ with untensed quantification and equality is defined, and it is demonstrated how to prove that a particular NL sentence cannot be expressed in $\mathbf{K}_t^{FO}$ through non-preservation under FOL path bisimulation. It is also shown that $\mathbf{K}_t^{FO}$ plus **now** and **then** is still strictly less expressive than $\mathbf{HL}(\downarrow, @)$, which is itself much less expressive than the strong systems that were claimed to be needed for NL translation. Thus the paper provides strict lower and upper bounds on the expressivity of the translation language forced by Kamp-Vlach NL sentences, and the upper bound turns out to be much lower than was widely believed in the linguistics community.

The current consensus in the linguistic community with respect to the debate about the exact expressive power needed for the temporal system of natural language (NL) holds that natural language requires the expressivity of logic with full first-order quantification over times. This is justified by referring to Cresswell's argument in [2].

[2]'s original argument for times points to two sources of evidence: first, to what we can observe about implicit time quantification, and second, to sentences of the form "*There were times when ...*" where reference to times is explicit. I do not consider the second kind of evidence in the current paper: it is clear that natural language is capable of expressing quite complicated quantificational constructions when the objects of quantification may be named; had it been otherwise, we would hardly be able to do mathematics. Natural language can express

[*] This paper has benefited greatly from my discussions with Salvador Mascarenhas, and from the comments of three anonymous WoLLIC reviewers. Remaining errors are, of course, mine alone.

L. Beklemishev and R. De Queiroz (Eds.): WoLLIC 2011, LNAI 6642, pp. 293–309, 2011.

complex statements about mathematical objects requiring very strong logics, and it would be surprising if the noun *time* would be more restricted than, say, the noun *number* in that respect. What is more interesting is what level of expressivity is required by the temporal system implicit in NL. The implicit temporal arguments cannot be assumed without a serious argument to be exactly the same "times" which a noun *time* could denote, so we need to investigate the temporal system on its own.

Regarding the implicit quantification over times (and worlds), Cresswell's argument goes roughly like this: first, NL has to be translated into a logic at least as strong as his logic storing a sequence of evaluation indices and including operators allowing to shift the evaluation of a subformula to any formerly introduced index. (If that sounds confusing, wait a little until we define a Cresswell-style system **Cr** shortly below.) The second step is to show that Cresswell's primary language $\mathcal{L}^*$ is equivalent to language $\mathcal{L}^+$, including operators equivalent to the more standard by now hybrid $\downarrow$ and @. $\mathcal{L}^+$, in its turn, is equivalent to a language with unrestricted universal quantification over times $\mathcal{L}$. A reader familiar with hybrid-logical research would become very surprised by the latter claim, of course, but the claim is explained by the fact that the basic modal language which Cresswell builds his other languages upon includes a universal modality not restricted by an accessibility relation ($\Box$ in his notation, with the usual R-restricted box being written L.)[1] Universal modality combined with $\downarrow$ and @ easily allow to express unrestricted universal quantification over times.

Using [1]'s notation for Cresswell's $\mathcal{L}^+$ operators, and A for universal modality, [2] defines $\forall u.\alpha$ as $\downarrow w.A\downarrow u.@w.\alpha$. Unfortunately, this easy exercise was misperceived by the linguistic community as a proof that NL requires full first-order quantification over times in the translation language.[2]

To more properly address the question of NL's expressive power, one should start with a mild basic language. This is what we will do in this paper, adopting the simple temporal language $\mathbf{K}_t$ as our basic language. Consider the examples which motivate that study in the first place, by demonstrating that $\mathbf{K}_t$ itself is less expressive than needed:

A child was born which will be ruler of the world. (1)

One day, all persons now alive will be dead. (2)

The problem with (1) is that *will* in the embedded sentence is in the scope of the past tense operator in the matrix clause, and yet shifts the interpretation to a point later than the *present*, not the past moment at which the child was born. Any compositional translation — one that preserves the relative scope between the past tense and the embedded future tense — must use some device to allow

[1] For Cresswell, the inclusion of that operator into the basic language is philosophically grounded.

[2] To give just one example of fine linguistic work which shares the mistake, let me cite [6, p. 44]: "Cresswell 1990 showed systematically that the full power of overt quantification over times and possible worlds is needed to handle modal and temporal talk in natural language."

for that. But at least there is an equivalent sentence of the standard temporal language: $\exists x : P(b(x)) \wedge F(rw(x))$.[3] In (2), the problem is worse: we either need some special device allowing us to get back to the present from under a future tense operator, or... there is no way to express (2) at all![4] Or, at least, so the conventional wisdom tells us; there is no formal proof that I know of. The success in dealing with (2) will be our measure of adequacy for translation logics: it is the "strongest" kind of examples used historically to argue for greater expressive power for natural language. There may be other types of examples requiring even greater expressivity, but I leave that for future research.

The plan of the paper is as follows. Section 1 builds the foundation for the main results, examining simpler propositional systems. We introduce the propositional **Cr**, which is essentially $\mathbf{K}_t$ enriched by operators **now** and $\mathbf{then}_k$ that shift the evaluation index of their scope to some index introduced earlier. We provide a translation from **Cr** to $\mathbf{K}_t$, not an unexpected result (see, e.g., [5] for a recent, slightly less general similar translation), and translations between **Cr** and $\mathbf{K}_t + {\downarrow} + @$. The landscape becomes more interesting once we move to the FO variants of the systems discussed, in Section 2. We introduce the FO systems $\mathbf{Cr}^{FO}$ and (implicitly) $\mathbf{K}_t^{FO}$, define the notion of FOL path bisimulation and prove that $\mathbf{K}_t^{FO}$ is invariant under it. Then we demonstrate that some formulas of $\mathbf{Cr}^{FO}$ are not preserved under such bisimulation, using for that the $\mathbf{Cr}^{FO}$ translation of the NL example (2). With the help from an earlier established connection with hybrid languages, we show that $\mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + {\downarrow} + @$, and then by an easy argument that $\mathbf{K}_t^{FO} + {\downarrow} + @$ is strictly less expressive than $\mathbf{HL}^{FO}(\downarrow, @)$. The main two results are thus as follows: 1) the most complex Kamp-Vlach examples such as (2) require the power of $\mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + {\downarrow} + @$; 2) those languages are strictly between $\mathbf{K}_t^{FO}$ and $\mathbf{HL}^{FO}(\downarrow, @)$ in the expressivity hierarchy.

The use in one paper of both **Cr**, close to systems familiar to linguists and philosophers, and hybrid languages, more familiar to modal logicians, may make it seem more complex than the paper actually is. Nevertheless, I find it important to make an explicit connection between the two kinds of systems, hoping that it

[3] For sentences of this kind, there is never a problem with providing a non-compositional translation, a point which is sometimes rediscovered. For instance, [7, pp. 130-132] cites his treatment of (1)-style examples as an advantage of his particular system for translation of tense, while the triumph follows completely from translating the two clauses as conjuncts, instead of treating one as being in the scope of the other's tense. Of course, such treatment for a sentence like (1) raises the question of what to do with its counterpart where *will* is replaced with *would*. Verkuyl stipulates that the variables in his translation should get identified with each other in conjoined clauses if those all have the PAST operator of his system. But there exist examples showing that syntactic embedding is after all relevant for the dependency. E.g., in *Mary said that Sue came, and Ann told me that John would call* it cannot be that *would* is dependent on the moment of Mary's utterance, it can only be dependent on the moment of Ann's speech; [7] cannot account for this without further tweaking his system, and does not discuss such examples.

[4] See (5) for a translation of that example with the **now** operator, and (6) for a translation into a language with explicit FO quantification over times.

would help to narrow the currently rather wide gap between logical and linguistic research.

1 The Propositional Case: No Additional Expressive Power

We start by introducing the propositional variant of a Cresswell-style system.[5] The intuitive idea behind $\mathbf{then}_k$ operators is that they serve as translations for overt NL adverbs like *now* in (2).

Definition 1. *Wff-s ϕ of* **Cr** *have the following shape (PROP is the set of propositional variables):*
$\phi \ := \ PROP \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid F\phi \mid P\phi \mid \mathbf{then}_k\,\phi$[6] *for $k \in \omega$.*
$\bot$, $\vee$ and $\rightarrow$ are defined as usual, and $\mathbf{now} := \mathbf{then}_0$*.*

Definition 2. ***Depth of (temporal) embedding*** *$Dep(i)(\xi(\phi))$, where i is the position*[7] *of some subformula ϕ of ξ: 1) if there are no* $\mathbf{then}_k$ *operators on top of the i-th instance of ϕ in ξ, $Dep(i)(\xi(\phi))$:= the number of F and P operators scoping over ϕ; 2) otherwise, with some* $\mathbf{then}_k$ *being the lowest* **then** *operator with ϕ in its scope, $Dep(i)(\xi(\phi)) := k + n$ where n is the number of F and P operators scoping over ϕ, but below that* $\mathbf{then}_k$*.*

When it is clear from context which subformula ϕ is meant, we write simply $Dep(\xi(\phi))$.

Definition 3. *A* sentence *of* **Cr** *ϕ is a wff which obeys the following* **then***-rule: for any occurrence of* $\mathbf{then}_k\,\psi$ *in ϕ where i is the position of that occurrence, $k \leq Dep(i)(\xi)$.*

Note that defining the distinction between formulas and sentences on the basis of the **then**-rule is unusual. But despite appearances, our notion of sentences is quite close to the standard one, as we shall see once the semantics for **Cr** is

[5] It is different from the original system in [2] in two important aspects: first, Cresswell builds unindexed evaluation sequences, only uses the first member as the evaluation point, and employs permutation operators on sequences to place the right index into the first position. Our formulation uses indexed sequences instead of permutation operators, where the index determines the "current" member of the sequence used for evaluation. $\mathbf{then}_k$ operators shift that index to the k-th member of the sequence. It is easy to see that it does not make the system substantially different, but the syntax and semantics of our variant are somewhat streamlined, in my opinion. Second, our **Cr** is based upon the standard temporal language $\mathbf{K}_t$, not on a language to which universal modality is added, as Cresswell's actual system in [2] is. This is crucial, of course, for Cresswell's choice obscures the difference **then** operators would make if added to a weaker underlying language.

[6] The bracketing convention for $\mathbf{then}_k$ is the same as for $\neg$, F and P. E.g., $\mathbf{then}_k\,\phi \wedge \psi$ is an abbreviated form of $[\mathbf{then}_k(\phi)] \wedge \psi$.

[7] To define standard numbering of subformulas, fix some standard ordering on trees, and view ξ as a tree.

given. For the moment, let's just note that $\mathbf{then}_2\, p$ and $FF\,\mathbf{then}_0\, FF\,\mathbf{then}_5\, p$ are not sentences, by the **then**-rule.

A Kripke model for **Cr** is defined in the same way as for the classical Priorean temporal language $\mathbf{K}_t$, as $\langle W, R, V\rangle$, where W is a non-empty set of times, R is the earlier-than accessibility relation, and V, valuation for propositional variables, a function from set $PROP$ to $\mathcal{P}(W)$.

Furthermore, we say that an *indexed sequence* $\langle \rho, i\rangle$ is a denumerable sequence of times ρ with $i \in \omega$ pointing to the member serving as the current evaluation point. We denote by $\rho(n)$, $n \in \omega$, the n-th member of ρ. We write $(\rho_1 \sim^n \rho_2)$ as an abbreviation for $\forall m \in \omega : (m \neq n) \rightarrow (\rho_1(m) = \rho_2(m))$. After these preliminaries, we can define truth for **Cr** in models at sequences:

Definition 4. *The semantics of* **Cr**.

For a wff ϕ, truth at an indexed sequence is defined as follows.[8]

$$\begin{array}{ll}
M, \langle \rho, i\rangle \models q & \textit{iff } \rho(i) \in V(q) \\
M, \langle \rho, i\rangle \models \top & \textit{always} \\
M, \langle \rho, i\rangle \models \neg\phi & \textit{iff not } M, \langle \rho, i\rangle \models \phi \\
M, \langle \rho, i\rangle \models \phi \wedge \psi & \textit{iff } M, \langle \rho, i\rangle \models \phi \textit{ and } M, \langle \rho, i\rangle \models \psi \\
M, \langle \rho, i\rangle \models F\phi & \textit{iff there is } \rho' \sim^{i+1} \rho \textit{ s.t. } \rho(i) R \rho'(i+1) \textit{ and } M, \langle \rho', i+1\rangle \models \phi \\
M, \langle \rho, i\rangle \models P\phi & \textit{iff there is } \rho' \sim^{i+1} \rho \textit{ s.t. } \rho'(i+1) R \rho(i) \textit{ and } M, \langle \rho', i+1\rangle \models \phi \\
M, \langle \rho, i\rangle \models \mathbf{then}_k\, \phi & \textit{iff } M, \langle \rho, k\rangle \models \phi
\end{array}$$

For a sentence ϕ, $\quad M, \rho \models \phi$ *iff* $M, \langle \rho, 0\rangle \models \phi$.

Sentences ϕ and ψ are equivalent, $\phi \equiv \psi$, iff for any M and ρ their truth coincides.

It is easy to see that for each subformula ψ of a sentence ϕ, there exists a single designated index i such that the truth of ϕ depends on the truth of ψ at sequences $\langle \rho, i\rangle$. The structure of the sentence determines what that i is. For instance, the truth of $P\psi$ may only depend on the truth of ψ at sequences $\langle \rho, 1\rangle$, but never on, say, $\langle \rho, 0\rangle$ or $\langle \rho, 2\rangle$.

In a fixed M, the truth of ϕ with no **then** operators only depends on $\rho(0)$. In fact, **Cr** formulas with no **then** operators are just $\mathbf{K}_t$ formulas, with $\rho(0)$ serving as the evaluation point:

Proposition 1. *For a formula ϕ of* **Cr** *with no* **then** *operators,*

$M, t \models_{\mathbf{K}_t} \phi \quad$ *iff for any ρ s.t. $\rho(0) = t$,* $\; M, \rho \models_{\mathbf{Cr}} \phi$.

Proof. Obvious induction on formulas, since $\mathbf{K}_t$ and **Cr** clauses for classical operators are equivalent.

When we evaluate a formula ϕ of **Cr**, $\rho(0)$ is used like the evaluation point, and the other members of the sequence essentially form a variable assignment for $\mathbf{then}_k$ operators. E.g., the truth of $\mathbf{then}_4\, p$ depends entirely on what $\rho(4)$ is.

[8] For the F clause, it should be noted that if $\exists t : \rho(i) R t$, we can always build a suitable ρ' s.t. $\rho' \sim^{i+1} \rho$ by setting $\rho'(i+1)$ to t. So we could equivalently formulate the clause as "... iff there is t s.t. $\rho(i) R t$, and for the unique ρ' defined by $\rho' \sim^{i+1} \rho$ and $\rho'(i+1) := t$, we have $M, \langle \rho', i+1\rangle \models \phi$." Similarly for P.

But say we evaluate at ρ some $F\psi$. To determine its truth, we check if ψ is true at any $\rho' \sim^1 \rho$ s.t. $\rho'(1)$ is accessible from $\rho(0)$. Those ρ' can also be divided into the initial segment of up to $\rho'(1)$, which stores the evaluation indices used up till now when going down the chain of embedding, and the final segment starting at $\rho'(2)$ serving as an arbitrary variable assignment. The difference between the two portions is that the indices of the initial portion is restricted by the operators of the formula, while the indices in the tail may be arbitrary.

Consider some instance of $\mathbf{then}_k\, \psi$ in ϕ. Let ϕ be evaluated at ρ, and $\mathbf{then}_k\, \psi$, at ρ'. If $\rho'(k)$ is a state introduced by processing a higher F or P clause, $\mathbf{then}_k$ shifts the interpretation of ϕ to a previously used index. In essence, $\mathbf{then}_k$ behaves like a variable over times bound by the quantifier implicit in the F or P which introduced the relevant member of ρ'. On the other hand, if $\rho'(k)$ is the same as it was in the initial ρ (which happens if it was never accessed by an F or P clause), $\mathbf{then}_k$ behaves like a free variable, depending on the "variable assignment" of ρ without $\rho(0)$. Recall now the **then**-rule of Def. 3: it ensures exactly that $\mathbf{then}_k$ operators always depend on some higher F or P, and never on the members of the initial ρ except $\rho(0)$. In other words, if the **then**-rule is obeyed, all "variables" $\mathbf{then}_k$ are bound. Hence why it makes sense to say that ϕ is a sentence iff it obeys the **then**-rule.

It is useful to define the notion of formulas obeying the second half the **then**-rule:

Definition 5. *A wff ϕ is **proper** iff for each subformula of the form $\mathbf{then}_k\, \psi$ whose index in ϕ is i, if there is some $\mathbf{then}_l$ scoping over $\mathbf{then}_k\, \psi$, then $k \leq Dep(i)(\phi)$.*

E.g., $\mathbf{then}_3\, P\, \mathbf{then}_4\, q$ is proper, while $\mathbf{then}_3\, P\, \mathbf{then}_5\, q$ is not. A formula with only one **then** will always be proper. The notion is useful because if some formula is not proper, it cannot be a subformula of any sentence: it would offend the **then**-rule embedded into any formula.

Definition 6. *For a wff ϕ, assign to each $\mathbf{then}_k\, \psi$ subformula whose index is i a number m which is $\mathtt{Max}(k - Dep(i)(\phi),\ 0)$. The maximal such m is the **maximal lookup depth** m_ϕ of ϕ.*

The maximal lookup depth of ϕ is the number of F and P operators one must add on top of formula ϕ to get a sentence. E.g., for $\mathbf{then}_3\, P\, \mathbf{then}_4\, q$ it is 3, while for $P\, \mathbf{then}_6\, q$ it is 5. For a sentence ϕ, obviously $m_\phi = 0$, from the **then**-rule.

Definition 7. *It is **proper** to evaluate a wff ϕ at $\langle \rho, i \rangle$ (in any M) iff 1) ϕ is proper and 2) the maximal lookup depth $m_\phi \leq i$. $\langle \rho, i \rangle$ is then a **proper indexed sequence for** ϕ.*

It is easy to check that if $\langle \rho, i \rangle$ is proper for ϕ, then whether $M, \langle \rho, i \rangle \models \phi$ does not depend on that particular ρ after the i-th member. Note that if $\langle \rho, i \rangle$ is proper for ϕ, then $\langle \rho, (i+1) \rangle$ is also proper for ϕ. The smaller i in $\langle \rho, i \rangle$, the less formulas it will be proper for. In particular, sequences $\langle \rho, 0 \rangle$ are proper only for sentences. If ϕ is a sentence, then for any subformula ψ of ϕ, $m_\psi \leq Dep(j)(\phi)$, where j is ψ's number in ϕ, or else the **then**-rule would have been offended.

We can now define the conditions under which two formulas can be substituted for each other.

Definition 8. *Two proper formulas ϕ and ψ of* **Cr** *are* ***strictly equivalent****, or $\phi \sim \psi$, iff for all M and all proper $\langle \rho, i \rangle$, $\quad M, \langle \rho, i \rangle \models \phi \quad$ iff $M, \langle \rho, i \rangle \models \psi$*

Theorem 1. *Let $\xi^{[\phi \to \psi]}$ be the result of substituting all instances of ϕ in ξ with ψ-s.*

If $\phi \sim \psi \quad$ and for any position i of ϕ in ξ, $\mathtt{Max}(m_\phi, m_\psi) \leq Dep(i)(\xi)$,
then for any sentence ξ, $\quad \xi \equiv \xi^{[\phi \to \psi]}$.

Proof. Since ξ is a sentence and $\mathtt{Max}(m_\phi, m_\psi) \leq Dep(i)(\xi)$ for any position of ϕ in ξ and ψ in $\xi^{[\phi \to \psi]}$, $\xi^{[\phi \to \psi]}$ is also a sentence. Take some $\langle \rho, 0 \rangle$ that makes ξ true. Fix an instance of ϕ in ξ numbered j. The truth of ξ only depends on the truth of that instance at sequences with index $Dep(j)(\xi)$. Since ξ is a sentence, $m_\phi \leq Dep(j)(\xi)$, so for any ρ sequences $\langle \rho, Dep(j)(\xi) \rangle$ are proper for ϕ. As $m_\psi \leq Dep(j)(\xi)$, exactly the same sequences $\langle \rho, Dep(j)(\xi) \rangle$ are proper for ψ as well. From $\phi \sim \psi$, the truth of the j-th subformula of ξ always coincides with the truth of the j-th subformula of $\xi^{[\phi \to \psi])}$. By induction on instances of ϕ in ξ, at any ρ which makes ξ true, $\xi^{[\phi \to \psi])}$ is also true. The other direction is given by the same argument after we notice that $\xi = (\xi^{[\phi \to \psi]})^{[\psi \to \phi]}$. □

We will also need the following fact, which will allow us to get rid of $\mathbf{then}_k$ in the translation:

Proposition 2. *Let all instances of $(\mathbf{then}_k\, \phi)$ in sentence ξ have the depth of embedding of exactly k. Then $\xi \equiv \xi^{[(\mathbf{then}_k\, \phi) \to \phi]}$.*

Proof. If the depth of $\mathbf{then}_k\, \phi$ in ξ is always k, it means that only $(\mathbf{then}_k\, \phi)$'s truth at $\langle \rho, k \rangle$ sequences can influence the truth of ξ. But at those sequences, the truth of $\mathbf{then}_k\, \phi$ coincides with the truth of ϕ: from Def. 4, $M, \langle \rho, k \rangle \models \phi$ iff $M, \langle \rho, k \rangle \models \mathbf{then}_k\, \phi$. □

Theorem 1 and Proposition 2 give us the tools we will use in the translation. We will use the following two strict equivalencies (and **then**-less tautologies) to float any $\mathbf{then}_k$ subformula up in sentence ϕ to the position in the sentence where $\mathbf{then}_k$ can be safely deleted — to a position i s.t. $Dep(i)(\phi) = k$. In the schema (4) below, O stands for F or P.

$$\neg\, \mathbf{then}_k\, \phi \sim \mathbf{then}_k\, \neg \phi \tag{3}$$

$$O((\mathbf{then}_k\, \phi) \wedge \psi) \sim (\mathbf{then}_k\, \phi) \wedge O\psi \tag{4}$$

(3) is obvious from the definitions. For (4), let's resolve O to F for concreteness — the argument for P is parallel. Note that the maximal lookup depth of the left side is at least $k-1$, and that of the right side, at least k. We are only interested in sequences proper for both sides, so we only look at $\langle \rho, i \rangle$ with $k \leq i$.

The left side is true at a proper $\langle \rho, i \rangle$ iff there is some $t > \rho(i)$, and $(\mathbf{then}_k\, \phi \wedge \psi)$ is true at $\langle \rho', (i+1) \rangle$ with $\rho' \sim^{i+1} \rho$ and $\rho'(i+1) = t$. Which, in its turn, is true iff there exists such a $t > \rho(i)$ that ϕ is true at $\rho'(k)$, and ψ is true at t. From $k \leq i$, we get $k \leq (i+1)$ and thus $\rho'(k) = \rho(k)$, for ρ and ρ' differ only on $i+1$. Now let's check the truth of the right side assuming the left side is true. The first conjunct ($\mathbf{then}_k\, \phi$) of the right side is true iff ϕ is true at $\rho(k)$, so if the left side is true, the first conjunct is true. The second conjunct $F\psi$ is true iff there is a $t > \rho(i)$ s.t. ψ is true at t — and again, if the left side is true, there is such a t. So for any $\langle \rho, i \rangle$ proper for both sides, if the left side is true, the right side is also true. By similar reasoning, if the right side is true at a proper $\langle \rho, i \rangle$, the left side is also true.

Note the crucial role that the presence of ψ plays in (4): $O(\mathbf{then}_k\, \phi)$ is *not* substitutable for $\mathbf{then}_k\, \phi$ (consider $\rho(i)$ where $\Box\bot$ is true.)

Theorem 2. *There exists a truth-preserving translation from sentences of* **Cr** *to sentences of* $\mathbf{K}_t$*, namely, such function* Tr *that for any* ϕ *of* **Cr***,* $M, \rho \models_{\mathbf{Cr}} \phi$ *iff* $M, \rho(0) \models_{\mathbf{K}_t} Tr(\phi)$*.*

The proof, including the definition of such a translation, is given in Appendix B. Thus the propositional **Cr** is just an alternative notation for the standard temporal language: it does not add any more expressive power, just puts things together in a different order.

But before we finish with **Cr**, we make another connection, which must be almost obvious for reader familiar with hybrid logics: instead of **then** operators, we could have added to the basic temporal language $\mathbf{K}_t$ the hybrid $\downarrow$ and @, but without adding nominal atoms. In fact, $\mathbf{Cr} = \mathbf{K}_t + \downarrow + @$ (and both are equal in expressive power to simple $\mathbf{K}_t$.) Right now, it may seem as a pointless exercise to show that, but the connection to hybrid systems will prove useful when we turn to the predicate case, where $\mathbf{Cr}^{FO} \neq \mathbf{K}_t^{FO}$.

Definition 9. *The syntax of* $\mathbf{K}_t + \downarrow + @$*.*

Atoms of $\mathbf{K}_t + \downarrow + @$ *come from two sets: PROP for propositional variables p, q, ..., and NOM for variables over times, or nominal variables, u,v, ...*

$\phi \;:=\; PROP \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid F\phi \mid P\phi \mid \downarrow u.\phi \mid @u.\phi.$

Note that NOM symbols cannot be atomic formulas. If they could, we'd have a full hybrid system $\mathbf{HL}(\downarrow, @)$. Formulas of $\mathbf{K}_t + \downarrow + @$ are evaluated with respect to a point and an assignment function from NOM to the set of times. $\downarrow u.\phi$ stores the current evaluation point in variable u; $@u.\phi$ retrieves the point stored in u and evaluates its argument ϕ at it. If $@u$ is used without a prior $\downarrow u$, the truth of the whole formula depends on the assignment function. If all $@u$ are in the scope of $\downarrow u$ — if they are bound, — then the formula as a whole does not depend on the assignment function. We call a formula which does not depend on the assignment function a sentence.

Definition 10. *The semantics of* $\mathbf{K}_t + \downarrow + @$*.*

We give the clause for F *as the example of a standard clause, and clauses for* $\downarrow$ *and* @*:*

$M, g, w \models F\phi$ *iff there is* w' *s.t.* wRw' *and* $M, g, w' \models \phi$
$M, g, w \models {\downarrow}u.\phi$ *iff for* g' *s.t.* $g' \sim^u g$, $g'(u) := w$, $M, g', w \models \phi$
$M, g, w \models @u.\phi$ *iff* $M, g, g(u) \models \phi$

For a sentence ϕ*, we can write simply* $M, w \models \phi$*, as the choice of* g *does not matter.*

It is immediately obvious that $\mathbf{K}_t{+}{\downarrow}{+}@$ is very similar to **Cr**. In fact, the two can be translated into each other: for each sentence of one, we can build an equivalent sentence in the other. The translations, establishing $\mathbf{K}_t + {\downarrow} + @ \ = \ \mathbf{Cr} \ (= \mathbf{K}_t)$, are given in Appendix A.

2 The Predicate Case: More Expressive Power Is Needed

Converting **Cr** and $\mathbf{K}_t$ into predicate languages is straightforward, but opens up a number of design choices. We will work in a system with constant domain semantics, and untensed quantifiers and equality. The first choice is relatively innocent, as far as I can see: at least changing constant domains to non-constant domains with possible sharing of individuals by points does not seem to affect the proofs. The second choice of untensed quantifiers and equality is crucial, though. Tensed quantifiers seem to be a bad choice for modelling natural language quantification, see fn. 9. Untensed equality provides shelter from crashes caused by inexistence of some interesting individual at a point, see a remark on p. 309. For concreteness, I will give constant domain semantics in the definitions.

Our basic system $\mathbf{K}_t^{FO}$ will be the first-order (FO) temporal language with constant domain semantics and no individual constants. Formulas of $\mathbf{K}_t^{FO}$ are evaluated in a model at a point with respect to an assignment function h which provides values for individual variables. The system $\mathbf{Cr}^{FO}$ will be like $\mathbf{K}_t^{FO}$, but with the addition of **then**-operators, just as in the propositional case. For brevity, I only give the definitions for $\mathbf{Cr}^{FO}$ explicitly; those for $\mathbf{K}_t^{FO}$ are easy to deduce.

Definition 11. *The syntax of* $\mathbf{Cr}^{FO}$*:* $\phi \ := \ q(\bar{x}) \mid x = y \mid \forall x\phi \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid F\phi \mid P\phi \mid \ \mathbf{then}_{k\in\omega}\,\phi$.

A constant domain model for $\mathbf{Cr}^{FO}$ is $\langle W, D, R, V\rangle$, where D is the set of individuals, and V is a function from $W \times Pred_n \rightarrow Pow(D^n)$, where $Pred_n$ is the set of n-place predicate symbols.

Definition 12. *The semantics of* $\mathbf{Cr}^{FO}$*.*

I only give the clauses which are specific to the FO variant of **Cr***, and the clauses for* F *and* **then** *for comparison with propositional* **Cr***. Let* $h' \sim^x h$ *be a short notation for* $\forall y : y \neq x \rightarrow h'(y) = h(y)$*.*

$$\begin{array}{ll}
M,h,\langle\rho,i\rangle \models q(x_1,...,x_n) & \textit{iff } \langle h(x_1),...,h(x_n)\rangle \in V(\rho(i),q)\\
M,h,\langle\rho,i\rangle \models x=y & \textit{iff } h(x)=h(y)\\
M,h,\langle\rho,i\rangle \models \forall x\phi & \textit{iff for all } h' \textit{ s.t. } h' \sim^x h,\ \ M,h',\langle\rho,i\rangle \models \phi\\
M,h,\langle\rho,i\rangle \models F\phi & \textit{iff there is } \rho' \sim^{i+1} \rho \textit{ s.t. } \rho(i)R\rho'(i+1)\\
 & \quad \textit{and } M,h,\langle\rho',i+1\rangle \models \phi\\
M,h,\langle\rho,i\rangle \models \mathbf{then}_k\,\phi & \textit{iff } M,h,\langle\rho,k\rangle \models \phi
\end{array}$$

$\phi \in \mathbf{Cr}^{FO}$ is a sentence iff the following holds: $M,h,\langle\rho,0\rangle \models \phi$ iff for any ρ' s.t. $\rho(0)=\rho'(0)$ and for any h', $M,h',\langle\rho',0\rangle \models \phi$. Alternatively, a sentence obeys the **then***-rule in Def. 3, and with a fixed $\langle\rho,i\rangle$ denotes a constant function from assignments h to truth values.*

For a sentence ϕ, we can write simply $M,\rho \models \phi$, omitting h.

Adding $\mathbf{then}_k$ to the classical propositional temporal language $\mathbf{K}_t$ does not actually increase the expressive power of the resulting language. But a first-order system with $\mathbf{then}_k$ (our $\mathbf{Cr}^{FO}$) *is* more expressive than $\mathbf{K}_t^{FO}$. Recall the natural language example (2). It can be easily rendered as (5) in $\mathbf{Cr}^{FO}$[9], or equivalently as (6) in a two-sorted FO logic with explicit quantification over times. But a few attempts are usually enough to convince myself there is no equivalent sentence of $\mathbf{K}_t^{FO}$.

$$F[\forall x : \mathbf{then}_0(q(x)) \to r(x)] \tag{5}$$

$$\exists t' > t : [\forall x : q(x)(t) \to r(x)(t')] \tag{6}$$

Before we proceed, it is instructive to point out what makes (5) so complex. Consider the $\mathbf{K}_t^{FO}$ formula $\forall x : q(x) \to F(r(x))$. It checks for each x that is q at the evaluation point t whether x will ever be r. But it cannot check if there is any particular moment *all* such x-s will be r together.

On the formal side, [4] has shown that the **now** operator is ineliminable in a first-order system on the class of all frames by demonstrating a construction which shows there is a special formula in a language with **now** not equivalent to any $\mathbf{K}_t^{FO}$ formula. But to my knowledge, there is no technique for showing

[9] A reviewer asks if (5) correctly captures (2) when the domains are not constant. That depends on whether quantifiers are tensed or untensed. Consider first a system with tensed quantifiers — those ranging over all individuals belonging to their evaluation point. Suppose there is x_1 which is q at the initial point t, but does not exist at the point t' which witnesses that other x-s which were q at t. Suppose all other x-s which are q at t are r at t'. Intuitively, (2) is false at t: x_1 was q at the initial point, but is not r together with the others. Yet (5) is true if $\forall x$ is tensed and ranges over those x-s which exist at t'. This tensed quantifier semantics (wrong for the natural language (2)) can be rendered like this: $\exists t' > t : [\forall x \in t' : q(x)(t) \to r(x)(t')]$, where $x \in t'$ means "x exists at point t'".

But with untensed quantifiers, things are all right. Consider some x_1 which is q at t, but does not exist at t'. It will not be r at t', then, so the formula will be false. Now consider some x_2 which exists at t', but not at t. As it does not exist at t, $q(x_2)$ is false there, so whether x_2 is r or not at t' is irrelevant for the truth of the formula. So (5) amounts to (6), and is all right.

that a *particular* formula like (5) is inexpressible in $\mathbf{K}_t^{FO}$. Using the advances in modal logic made since the time of [4], it is easy to develop such a technique by defining a proper notion of bisimulation for $\mathbf{K}_t^{FO}$ and showing for formulas like 5 that they are not preserved under such bisimulation.

Definition 13. *Let M be a model, w a point in M, $a_1, ...a_n$, $n \in \omega$, individuals at point w.*

$M, w, \langle a_1, ...a_n \rangle \models \phi$ *iff* $M, w, h \models \phi$ *where* h *assigns* a_1 *to the lexicographically first free variable in* ϕ*, and so forth.*

$M, w \models \phi$ *iff there exists some* $\langle a_1, ...a_n \rangle$ *s.t.* $M, w, \langle a_1, ...a_n \rangle \models \phi$

Definition 14. Non-modal and modal theories of a tuple at a point:

$Th_{FOL}(M, w, \langle a_1, ...a_n \rangle) := \{\phi \mid \phi$ has no modal operators and $M, w, \langle a_1, ...a_n \rangle \models \phi\}$

$Th_{\mathbf{K}_t^{FO}}(M, w, \langle a_1, ...a_n \rangle) := \{\phi \mid \phi \in \mathbf{K}_t$ and $M, w, \langle a_1, ...a_n \rangle \models \phi\}$

Now we can make an obvious, and wrong, first stab at the notion of bisimulation for $\mathbf{K}_t^{FO}$:

Definition 15. *Non-empty relation E is a* ***FOL bisimulation*** *between two structures M and N iff: 1)* ***FOL harmony****: if wEw', then for any $\langle a_1, ...a_n \rangle \in w$, there is $\langle b_1, ...b_n \rangle \in w'$ s.t. $Th_{FOL}(M, w, \langle a_1, ...a_n \rangle) = Th_{FOL}(N, w', \langle b_1, ...b_n \rangle)$, and vice versa; 2)* ***Zig and Zag****: if Rwv and wEw', then $\exists v'$ s.t. vEv' and $R'w'v'$, and vice versa.*[10]

This is not too bad: all formulas not containing modal operators are preserved between two bisimilar points, so we cannot distinguish such points from the inside. Accessibility relations should also be parallel because of Zig and Zag. Since both conditions hold together, we cannot distinguish between the points we can arrive to by traveling along accessibility links by the internal theories of points we pass either.

So the only way something can go wrong is when we check something beyond simple non-modal theories at connected points. This is exactly what happens when we have quantifiers binding variables in scope of different modal operators. Consider these two models:

$M: \quad W_M = \{w_1 : q(a), \neg q(b); \quad w_2 : q(a), \neg q(b)\} \quad R_M = \{\langle w_1, w_2 \rangle\}$
$N: \quad W_N = \{v_1 : q(c), \neg q(d); \quad v_2 : \neg q(c), q(d)\} \quad R_N = \{\langle v_1, v_2 \rangle\}$

Both M and N have just two points, and two individuals living at each point. All four points have the same internal FO theory: $\exists x, y : q(x) \wedge \neg q(y)$. Zig and Zag hold for w_1 and v_1. So there is a FOL bisimulation E s.t. $w_1 E v_1$. Yet $\exists x : q(x) \wedge F(q(x))$ is true at w_1 and false at v_1. Obviously the truth of $\mathbf{K}_t^{FO}$ formulas is not preserved under FOL bisimulation as defined in Def. 15.

What went wrong? The formula $\exists x : q(x) \wedge F(q(x))$ checks the properties of the same object at two points, but the notion of FOL bisimulation does not see beyond one point when it comes to the theory of an individual. So we need to preserve not only the non-modal theories of individuals at bisimilar points, but also the modal theories of individuals, or, rather, of tuples of individuals.

[10] Note that the bisimulation notions given in Definitions 15 and 17 are implicitly relative to a fixed language, for it determines the theories of individual tuples at points.

Definition 16. *Let L be a modal language with modal operators of arity 1, and $O_\Diamond$ be the set of L's diamonds. An* accessibility path $\pi := O\pi \mid \Lambda$*, where Λ is the designated empty path.*

For w_1, w_2 points in L's model M, a non-empty path $\pi = \Diamond_{i_1}...\Diamond_{i_n}$ leads from w_1 to w_2 (in symbols, $w_1 \pi w_2$) iff there exist points v_{i_1}, ..., $v_{v_{n-1}}$ s.t. $w_1 R_{i_1} v_{i_1} ... v_{i_{n-1}} R_{i_n} w_2$. For any w, $w \Lambda w$.

To see what is the significance of the new notion, note that if there is a path $\pi_1 := FF$ between w_1 and w_2 in M, then the truth of some ϕ at w_2 guarantees that $M, w_1 \models FF\phi$.

From the point of view of $\mathbf{K}_t^{FO}$, two tuples of individuals $\langle a_1, ... a_n\rangle$ and $\langle b_1, ... b_n\rangle$ at a point w are indistinguishable if for each v reachable from w, there is a v' reachable from w by the same accessibility path s.t. $Th(M, v, \langle a_1, ..., a_n\rangle) = Th(M, v', \langle b_1, ..., b_n\rangle)$. Thus we replace FOL harmony with stronger FOL path harmony in the definition of bisimulation:

Definition 17. *Non-empty relation E is a* ***FOL path bisimulation*** *between two structures M and N iff: 1)* ***FOL path harmony****: If wEw', then for each $\langle a_1, ... a_n\rangle \in w$, there is a $\langle b_1, ... b_n\rangle \in w'$ s.t. for any path π and v : if $w\pi v$, there is a v' s.t. $w'\pi v'$ and $Th(M, v, \langle a_1, ... a_n\rangle) = Th(N, v', \langle b_1, ... b_n\rangle)$; and vice versa; 2)* ***Zig and Zag****: if Rwv and wEw', then $\exists v'$ s.t. vEv' and $R'w'v'$; and vice versa.*

This notion is still modal (that is, relatively weak) in nature: for instance, we can easily build a reflexive and an irreflexive models FOL-path-bisimilar to each other. But $\mathbf{K}_t^{FO}$ formulas like $\exists x : q(x) \wedge F(r(x))$ will be preserved because of the new condition of FOL path harmony.

Theorem 3. *If E is a FOL path bisimulation between M and N and wEw', then for all $\phi \in K_t^{FO}$, $M, w \models \phi$ iff $N, w' \models \phi$.* The proof is given in Appendix B.

So far so good: FOL path bisimulation is strong enough to preserve the truth of $\mathbf{K}_t^{FO}$. But what if it preserves the truth of formulas with **then** as well? We give an example showing it is not so.

We first define the model M as follows: W in countably infinite and consists of w, u, and $v_0, v_1, v_2, ...$ R is such that wRu, wRv_0, wRv_1, so wRx holds for every point other than w, and no other pair of points is in R. The domain of individuals consists of two countably infinite sets: $D := \{a_0, a_1, ...; b_0, b_1, ...\}$. There is only one 1-place predicate symbol q. At every point, all b-s are not q. At w and u, all a-s are q. At all other points, one of the a-s is $\neg q$: at v_0, it is a_0, at v_1, it is a_1, and so forth. To finish the construction, we take the restriction of M to just w and v-s, without the u point, and call that restriction N.

The difference between M and N is that only in M the $\mathbf{Cr}^{FO}$ formula (5) is true at w: M contains u, which is the only point F-reachable from w where all x-s that are q in w are q as well. N does not contain u, and at every other F-reachable point, one of the a-s is not q, though it was q at the starting point w, so in N at w, (5) is false.

But $\langle M, w\rangle$ and $\langle N, w\rangle$ are FOL path bisimilar, which means they are indistinguishable in $\mathbf{K}_t^{FO}$. A possible FOL path bisimulation E may be defined as

wEw, uEv_0, v_0Ev_1, and so forth. Zig and Zag are clearly satisfied. Let's check FOL path harmony. Take some a_i at w. There are two sorts of points accessible from w in M, as far as a_i is concerned: those there $q(a_i)$ (most of the points), and those where $\neg q(a_i)$ (just v_i). From the perspective of a single a_i, it does not matter if u is in the model or not: it is just one of the countably infinite points where $q(a_i)$ holds. For any finite tuple of a_i, it would be the same. b-s are even simpler. So FOL path harmony holds, and E is a FOL path bisimulation, therefore there is no $\mathbf{K}_t^{FO}$ formula able to distinguish between w in M and in N. Yet (5) can. Thus $\mathbf{K}_t^{FO} \subsetneq \mathbf{Cr}^{FO}$: the latter is strictly more expressive.

After we characterized $\mathbf{Cr}^{FO}$ from below, we also give an upper bound, and for that we will use the direct connection to hybrid systems which was seemingly pointless in the previous section. In fact, essentially the same translations as in the propositional case show that $\mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + \downarrow + @$, see Appendix A. And we can exploit that fact and compare the relative power of $\mathbf{K}_t^{FO} + \downarrow + @$ vs. $\mathbf{HL}^{FO}(\downarrow, @)$, the full hybrid system with $\downarrow$ and @.

Definition 18. *The syntax of* $\mathbf{HL}^{FO}(@, \downarrow)$*.*

$\phi \ := \ q(x_1, ..., x_{n\in\omega}) \mid x = y \mid NOM \mid \top \mid \neg\phi \mid \phi \wedge \phi \mid F\phi \mid P\phi \mid \forall x\phi \mid \downarrow u.\phi \mid @u.\phi.$

Definition 19. *The semantics of* $\mathbf{HL}^{FO}(@, \downarrow)$*.*

As usual, we only give the crucial clauses, and illustrate how a couple of familiar clauses get modified. g is the assignment function for symbols in NOM (cf. the definition of $\mathbf{K}_t + \downarrow + @$*); h, for individual variables x.*

$M, g, h, w \models u$	*iff* $g(u) = w$
$M, g, h, w \models \downarrow u.\phi$	*iff for* g' *s.t.* $g' \sim^u g$, $g'(u) = w$, $M, g', h, w \models \phi$
$M, g, h, w \models @u.\phi$	*iff* $M, g, h, g(u) \models \phi$
$M, g, h, w \models \forall x\phi$	*iff for all* h' *s.t.* $h' \sim^x h$, $M, g, h, w \models \phi$
$M, g, h, w \models F\phi$	*iff there is* w' *s.t.* wRw' *and* $M, g, h, w' \models \phi$

Obviously, the formulas of $\mathbf{K}_t^{FO} + \downarrow + @$ are also formulas of $\mathbf{HL}^{FO}(@, \downarrow)$ with the same interpretation, so $\mathbf{K}_t^{FO} + \downarrow + @ \ \subseteq \ \mathbf{HL}^{FO}(@, \downarrow)$.

For an easy way to show that $\mathbf{HL}^{FO}(@, \downarrow)$ is strictly more expressive than $\mathbf{K}_t^{FO} + \downarrow + @$, let's restrict our attention to "pure" formulas: formulas with only modal operators, nominals, and nominal binders. The pure formulas of $\mathbf{K}_t^{FO} + \downarrow + @$ are also formulas of $\mathbf{K}_t + \downarrow + @$. By the results in Section 1 and Appendix A, all formulas of $\mathbf{K}_t + \downarrow + @$ have $\mathbf{K}_t$ equivalents. So all pure formulas of $\mathbf{K}_t^{FO} + \downarrow + @$ are expressible in $\mathbf{K}_t$. But clearly not all formulas of $\mathbf{HL}(@, \downarrow)$ are. For instance, $\downarrow u.F\neg u$ expresses irreflexivity of the current point, inexpressible in $\mathbf{K}_t$.

It is easy to show that no amount of impure stuff we can add to pure $\mathbf{K}_t^{FO} + \downarrow + @$ can get us to express properties like irreflexivity. We thus have the following proper inclusions with respect to expressive power:

$$\mathbf{K}_t^{FO} \quad \subsetneq \quad \mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + \downarrow + @ \quad \subsetneq \quad \mathbf{HL}^{FO}(@, \downarrow) \tag{7}$$

3 Conclusion

Language $\mathbf{HL}^{FO}(@,\downarrow)$ is much less powerful than two-sorted FOL with full quantification over times or Cresswell's basic modal language: it cannot, for instance, express universal modality (see [1]). But the kind of examples which are cited in the linguistic and philosophical literature to motivate the choice of such very expressive languages, upon closer examination, turn out to require only the power of $\mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + \downarrow + @$, strictly less expressive than $\mathbf{HL}^{FO}(@,\downarrow)$. Returning to the question in the title of this paper, it turns out that though we do need more power than the classical Priorean system has for NL translation, we do not seem to need much more. This is close to the conclusion reached by [5], and it should be studied how the current results are related to Meyer's, which I leave for future work.

It must be stressed that the investigation should not stop here. We have only examined the examples usually used in the discussions of NL's expressive power. It may be that some other kinds of examples require more power. In particular, it may be that nominals should be allowed as atoms (cf. [3]'s concern about nominal *now* as in *Now will never happen again*, which is different from the adverbial *now* we discussed in this paper.) Further study is needed, and it will depend on its results whether it will be practically interesting to further investigate the formal properties of **Cr**-style systems.

References

1. Blackburn, P., Seligman, J.: Hybrid languages. Journal of Logic, Language and Information 4, 251–272 (1995)
2. Cresswell, M.: Entities and Indices. Kluwer, Dordrecht (1990)
3. Goranko, V.: Hierarchies of modal and temporal logics with reference pointers. Journal of Logic, Language and Information 5(1), 1–24 (1996)
4. Kamp, H.: Formal properties of "now". Theoria 37, 227–273 (1971)
5. Meyer, U.: 'now' and 'then' in tense logic. Journal of Philosophical Logic 38(2), 229–247 (2009)
6. Schlenker, P.: A plea for monsters. Linguistics and Philosophy 26, 29–120 (2003)
7. Verkuyl, H.: Binary tense. CSLI lecture notes, vol. 187. CSLI Publications, Stanford (2008)

A Translations between Cr and $\mathbf{K}_t + \downarrow + @$

Proposition 3. *Translation from* **Cr** *into* $\mathbf{K}_t + \downarrow + @$*:*

Build the $\mathbf{K}_t + \downarrow + @$ *sentence* ϕ' *as follows, applying for each* F*,* P *and* $\mathbf{then}_k$*-headed subformula the following operation exactly once: for each* $F(\psi)$ *or* $P(\psi)$ *subformula, replace* ψ *with* $\downarrow u_{Dep(i)(\phi)}.\psi$*, where* i *is* ψ*'s position in* ϕ*; for* $\mathbf{then}_k\,\psi$*, replace* $\mathbf{then}_k\,\psi$ *with* $@u_k.\psi$*.*

$$M,\rho \models_{\mathbf{Cr}} \phi \quad \textit{for any } \rho \textit{ s.t. } \rho(0) = t \quad \textit{iff} \quad M,t \models_{\mathbf{K}_t+\downarrow+@} \phi'$$

Proof. Immediately follows from the observation that at each evaluation level, the assignment function g is mimicking the initial segment of ρ used by that time, with $g(u_j)$ equal to $\rho(j)$. □

Proposition 4. *Translation from* $\mathbf{K}_t + \downarrow + @$ *to* $\mathbf{Cr}$.

In the $\mathbf{K}_t + \downarrow + @$ *sentence* ϕ*, for each instance of* $\downarrow u$ *operator, do the following: 1) compute* $a_i := Dep(i)(\phi)$ *for* i *the position of* $\downarrow\psi$*,*[11] *2) for each* $@u$ *bound by that instance of* $\downarrow u$*, replace it with* $\mathbf{then}_{a_i}$*; 3) when there are no more bound* $@u$*, delete* $\downarrow u$*.*

$$M, t \models_{\mathbf{K}_t+\downarrow+@} \phi \quad \textit{iff} \quad \textit{for any } \rho \textit{ s.t. } \rho(0) = t \;\; M, \rho \models_{\mathbf{Cr}} \phi'$$

Proof. Similarly to the previous proposition, the translation now emulates the assignment function of $\mathbf{K}_t + \downarrow + @$ using the evaluation sequence of $\mathbf{Cr}$, so in the end each subformula gets evaluated relative to the same tuple of points in ϕ and in ϕ'. □

The translations are so straightforward because functions g of $\mathbf{K}_t + \downarrow + @$ and sequences ρ of $\mathbf{Cr}$ represent the same information in them, only recorded in different form. In fact, they are insensitive to what gets evaluated at those points, carrying over to the FO variants:

Proposition 5. *Translations defined in Propositions 3 and 4 preserve truth when applied to* $\mathbf{Cr}^{FO}$ *and* $\mathbf{K}_t^{FO} + \downarrow + @$.

Proof. The same argument sketched for Propositions 3 and 4 above. □

Proposition 5 makes our life easier when it comes to comparison between $\mathbf{Cr}^{FO}$ and $\mathbf{HL}^{FO}(\downarrow, @)$, allowing us to easily establish that $\mathbf{Cr}^{FO} = \mathbf{K}_t^{FO} + \downarrow + @ \subsetneq \mathbf{HL}^{FO}(\downarrow, @)$.

B Proofs of Theorems 2 and 3

Proof of Theorem 2. We define Tr from $\mathbf{Cr}$ to $\mathbf{K}_t$ such that $M, \rho \models_{\mathbf{Cr}} \phi$ iff $M, \rho(0) \models_{\mathbf{K}_t} Tr(\phi)$:

1. For each subformula $\mathbf{then}_k\,\psi$ at position j s.t. $Dep(j)(\phi) = k$, apply Proposition 2 to substitute $\mathbf{then}_k\,\psi$ with just ψ.[12]
2. For each subformula $\mathbf{then}_k\,\psi$ at position j with $Dep(j)(\phi) \neq k$ and not immediately embedded under another $\mathbf{then}_l$, do the following:
   (a) Fix the "Boolean connective neighborhood" ζ of $\mathbf{then}_k\,\psi$ — the scope of the closest F or P operator having the position j in its scope.
   (b) Transform ζ into disjunctive normal form.

[11] It is straightforward to define $Dep(i)(\phi)$ for $\mathbf{K}_t + \downarrow + @$ in the similar manner to that which we used for $\mathbf{Cr}$, counting the number of dominating F and P operators between the subformula and either the top level or the closest higher $@u$ operator.

[12] I sloppily write ϕ in the definition, instead of using a temporary formula storing the intermediate result.

(c) Apply the modal tautology $F(\phi \vee \psi) \leftrightarrow F(\phi) \vee F(\psi)$ (or similarly for P) to get each member of the disjunctive normal form immediately under the modal operator. We get $\bigvee_i F(\xi_i)$ or $\bigvee_i P(\xi_i)$ where each ξ_i is a conjunction.

(d) Take all $F(\xi_i)$ or $P(\xi_i)$ with $\mathbf{then}_k\, \psi$ being in ξ_i and not under any F or P in it. For each such ξ_i, if there is negation on top of $\mathbf{then}_k$, use (3) to get the negation inside. Then use (4) to get $\mathbf{then}_k$ past the F or P operator (if $\xi_i = \mathbf{then}_k\, \psi$, apply the tautology $\psi \leftrightarrow (\psi \wedge \top)$ to make (4) applicable.) After all ξ_i are taken care of, all "descendants" of the original $\mathbf{then}_k\, \psi$ in position j are at depth $Dep(j)(\phi) - 1$.

3. If there are no **then** operators, halt. If there are some left, go back to step 1.

Any particular $\mathbf{then}_k\, \psi$ at initial position j in ϕ is guaranteed to be eliminated after some number of iterations. For any ψ, it will be that $k \leq m_{(\mathbf{then}_k\, \psi)} \leq Dep(j)(\phi)$. If $k = Dep(j)(\phi)$, $\mathbf{then}_k$ can be eliminated right away by Step 1. If $k < Dep(j)(\phi)$, unless it is in the immediate scope of another **then**, Step 2 will move it higher to depth $Dep(j)(\phi) - 1$. So eventually our $\mathbf{then}_k\, \psi$ will either hit the depth k where it can be eliminated, or will get stuck immediately under another $\mathbf{then}_l$.[13] But the same reasoning applies to that $\mathbf{then}_l$ as well: it should either be eliminated or get stuck at some step. But one **then** will have to be the highest one in any ϕ, so it will never get stuck. Then all other **then**-s which got stuck behind it will be able to rise further, and so forth. By induction on **then** operators in ϕ, we can eliminate all of them after a finite number of steps.

As all transformations preserve equivalence, after the translation halts, we have a **then**-less sentence $Tr(\phi)$ true at the very same ρ in M where ϕ is true. $Tr(\phi)$ is a $\mathbf{K}_t$ formula as well, and by Proposition 1, $M, \rho \models_{\mathbf{Cr}} \phi$ iff $M, \rho(0) \models_{\mathbf{K}_t} Tr(\phi)$. □

Note that the translation defined in the proof is EXPTIME because of the normalization step 2b. It would be interesting to learn if a more efficient translation can be defined which avoids normalization.

Proof of Theorem 3. The main work is done by the following lemma:

Lemma 1. *If two tuples $A := \langle a_1, ..., a_n \rangle$ in M at w and $A' := \langle a'_1, ..., a'_n \rangle$ in N at w' are in FOL path harmony, and wEw', then $Th_{\mathbf{K}_t^{FO}}(M, w, A) = Th_{\mathbf{K}_t^{FO}}(N, w', A')$.*

Proof of Lemma 1. Suppose there exist A and A' falsifying the lemma. Fix some ϕ for which $M, w, \langle a_1, ..., a_n \rangle \models \phi$, but $N, w', \langle a'_1, ..., a'_n \rangle \not\models \phi$. In this case, we can always pin down a minimal subformula of ϕ which contradicts the assumption that A and A' are in FOL path harmony.

ϕ may be viewed as several minimal subformulas ψ combined with the help of Boolean connectives. At least one of those ψ must also be such that $M, w, \langle a_1, ..., a_n \rangle \models \psi$, but $N, w', \langle a'_1, ..., a'_n \rangle \not\models \psi$, so we fix one. If ψ is atomic, we

[13] In this case, it can only be that $k \leq l$. Otherwise $\mathbf{then}_k\, \psi$ would have been eliminated before getting stuck.

immediately derive a contradiction because of harmony between A and A' at w and w'.

If $\psi = \Diamond\xi$, since $N, w', \langle a'_1, ..., a'_n\rangle \not\models \psi$, there must be no v' s.t. $w'Rv'$ where ξ is true for A'. As $M, w, \langle a_1, ..., a_n\rangle \models \psi$, we can find some v for which wRv and $M, v, \langle a_1, ..., a_n\rangle \models \xi$. But by Zig, $\exists v' : w'Rv' \wedge vEv'$. We now have vEv', and $M, v, \langle a_1, ..., a_n\rangle \models \xi$, and $N, v', \langle a'_1, ..., a'_n\rangle \not\models \xi$. We now examine A, A', v, v' and ξ as we did for A, A', w, w', and ϕ, but ξ has at least one operator less than ϕ.

Finally, consider $\psi = \forall x\xi$. Observe that if A and A' are in FOL path harmony, for any b there is b' s.t. $\langle a_1, ..., a_n, b\rangle$ and $\langle a'_1, ..., a'_n, b'\rangle$ are also in FOL path harmony, and vice versa.[14] Now by the hypothesis that $N, w', \langle a'_1, ..., a'_n\rangle \not\models \forall x\xi$, for some c', $N, w', \langle a'_1, ..., a'_n, c'\rangle \not\models \xi$. But there must be c''s counterpart c s.t. $\langle a_1, ..., a_n, c\rangle$ and $\langle a'_1, ..., a'_n, c'\rangle$ are in FOL path harmony. Again, we can now examine two tuples in FOL path harmony $\langle a_1, ..., a_n, c\rangle$ and $\langle a'_1, ..., a'_n, c'\rangle$ s.t. one of them makes ξ true, but not the other, and ξ is again smaller than ϕ was. So we return to the initial situation, but having a smaller formula.

As formulas are finite, we will eventually get down to the level of atomic formulas, and it will become evident that A and A' cannot be in FOL path harmony if their $\mathbf{K}_t^{FO}$ theories are different. $\dashv$

The rest is easy. Suppose for an arbitrary $\phi \in \mathbf{K}_t^{FO}$, $M, w \models \phi$. Fix some $\langle a_1, ..., a_n\rangle$ witnessing that. By FOL path harmony, if there is a FOL-path-bisimulation E between M and N and wEw', then there is a corresponding tuple $\langle a'_1, ..., a'_n\rangle$ at w' in N. By the lemma above, $Th_{\mathbf{K}_t^{FO}}(M, w, \langle a_1, ..., a_n\rangle) = Th_{\mathbf{K}_t^{FO}}(N, w', \langle a'_1, ..., a'_n\rangle)$, so from $\phi \in Th_{\mathbf{K}_t^{FO}}(M, w, \langle a_1, ..., a_n\rangle)$ we immediately get that ϕ is also in $Th_{\mathbf{K}_t^{FO}}(N, w', \langle a'_1, ..., a'_n\rangle)$, and thus $N, w' \models \phi$. □

How much does this proof rely on the choices for the FO component of our language? As long as quantifiers and equality are untensed, as far as I can see, it does not matter what kind of semantics for the domains at different points we give. Consider a tuple $\langle a_1, a_2\rangle$ such that only a_1 is in the domain of some w. $Th_{FOL}(M, w, \langle a_1, a_2\rangle)$ will still be well-defined; it will contain at least formulas of the sort $\phi(x) \wedge y = y$. If equality is untensed, a point-internal theory will contain formulas including tautological statements like $y = y$ regarding the individuals from the tuple non-existing at the point, but no non-trivial statements regarding the corresponding variables. The proof above thus does not imply existence of the members of tuples at the considered points.

[14] Suppose it were not so. Then there exist π and modal-less ζ s.t. $\pi\zeta$ is true for $\langle a_1, ..., a_n, b\rangle$ at w, but there is no b' s.t. $\pi\zeta$ is true for $\langle a'_1, ..., a'_n, b'\rangle$ at w'. But then $M, w, \langle a_1, ..., a_n\rangle \models \pi\exists x\zeta$, but $N, w', \langle a'_1, ..., a'_n\rangle \not\models \pi\exists x\zeta$, thus ζ is in some π point-internal theory of A, but false at all π-reachable points for A'. That contradicts the assumption A and A' are in FOL path harmony.

Author Index