Lecture Notes in Computer Science 5407

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Sergei Artemov Anil Nerode (Eds.)

Logical Foundations of Computer Science

International Symposium, LFCS 2009
Deerfield Beach, FL, USA, January 3-6, 2009
Proceedings

Springer

Volume Editors

Sergei Artemov
CUNY Graduate Center, Computer Science
365 Fifth Avenue, New York, NY 10016, USA
E-mail: sartemov@gc.cuny.edu

Anil Nerode
Cornell University, Department of Mathematics
545 Malott Hall, Ithaca, NY 14853, USA
E-mail: anil@math.cornell.edu

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.4, F.3, I.2.3, I.2.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-92686-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-92686-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12587964 06/3180 5 4 3 2 1 0

Preface

The Symposium on Logical Foundations of Computer Science series provides a forum for the fast-growing body of work in the logical foundations of computer science, e.g., those areas of fundamental theoretical logic related to computer science. The LFCS series began with "Logic at Botik," Pereslavl-Zalessky, 1989, which was co-organized by Albert R. Meyer (MIT) and Michael Taitslin (Tver). After that, organization passed to Anil Nerode.

Currently LFCS is governed by a Steering Committee consisting of Anil Nerode (General Chair), Stephen Cook, Dirk van Dalen, Yuri Matiyasevich, John McCarthy, J. Alan Robinson, Gerald Sacks, and Dana Scott.

The 2009 Symposium on Logical Foundations of Computer Science (LFCS 2009) took place in Howard Johnson Plaza Resort, Deerfield Beach, Florida, USA, during January 3–6. This volume contains the extended abstracts of talks selected by the Program Committee for presentation at LFCS 2009.

The scope of the symposium is broad and contains constructive mathematics and type theory; automata and automatic structures; computability and randomness; logical foundations of programming; logical aspects of computational complexity; logic programming and constraints; automated deduction and interactive theorem proving; logical methods in protocol and program verification; logical methods in program specification and extraction; domain theory logics; logical foundations of database theory; equational logic and term rewriting; lambda and combinatory calculi; categorical logic and topological semantics; linear logic; epistemic and temporal logics; intelligent and multiple agent system logics; logics of proof and justification; nonmonotonic reasoning; logic in game theory and social software; logic of hybrid systems; distributed system logics; mathematical fuzzy logic; system design logics; other logics in computer science.

We thank the authors and reviewers for their contributions. We acknowledge the support of Cornell University, the Graduate Center of the City University of New York, and Florida Atlantic University.

We are grateful to Evan Goris for preparing this volume for Springer.

October 2008

Anil Nerode
Sergei Artemov

Organization

Steering Committee

Stephen Cook	University of Toronto, Canada
Yuri Matiyasevich	Steklov Mathematical Institute, St. Petersburg, Russia
John McCarthy	Stanford University, USA
Anil Nerode	Cornell University, USA (General Chair)
J. Alan Robinson	Syracuse University, USA
Gerald Sacks	Harvard University, USA
Dana Scott	Carnegie-Mellon University, USA
Dirk van Dalen	Utrecht University, The Netherlands

Program Committee

Sergei Artemov	CUNY Graduate Center, New York, USA (PC Chair)
Matthias Baaz	Vienna University of Technology, Austria
Andreas Blass	University of Michigan, Ann Arbor, USA
Samuel Buss	University of California, San Diego, USA
Rod Downey	Victoria University of Wellington, New Zealand
Ruy de Queiroz	Universidade Federal de Pernambuco, Recife, Brazil
Petr Hájek	Institute of Computer Science, Prague, Czech Republic
Denis Hirschfeldt	University of Chicago, USA
Rosalie Iemhoff	Utrecht University, The Netherlands
Bakhadyr Khoussainov	The University of Auckland, New Zealand
Yves Lafont	Institut de Mathématiques de Luminy, Marseille, France
Daniel Leivant	Indiana University, Bloomington, USA
Robert Lubarsky	Florida Atlantic University, Boca Raton, USA
Victor Marek	University of Kentucky, Lexington, USA
Franco Montagna	Università Degli Studi di Siena, Italy
Anil Nerode	Cornell University, USA
Michael Rathjen	University of Leeds, UK
Philip Scott	University of Ottawa, Canada
Alex Simpson	University of Edinburgh, UK
Anatol Slissenko	Université Paris 12, France
Alasdair Urquhart	University of Toronto, Canada
Rineke Verbrugge	University of Groningen, The Netherlands

Additional Reviewers

Zena Ariola
Eugene Asarin
Vince Barany
Daniele Beauquier
Arnold Beckmann
Marta Bilkova
Manuel Bodirsky
Alexander Bolotov
Agata Ciabattoni
Robin Cockett
Veronique Cortier
Giuseppe De Giacomo
Simon Doherty
Mel Fitting
Joerg Flum
Gaelle Fontaine
Lou Goble
Pieter Hofstra
Lukasz Kaiser
Barteld Kooi
Frederic Koriche
Stephane Lengrand
John Longley
Harry Mairson
Joao Marques-Silva
Damiano Mazza
George Metcalfe
Marius Minea
Angelo Montanari
Ben Moszkowski
Luca Paolini
Alexander Rabinovich
Brian Redmond
Jan Reimann
Gerard Renardel de Lavalette
Bryan Renne
Robert Rettinger
Robert Rosebrugh
Paul Rowe
Paul Roziere
Sasha Rubin
Vladimir Rybakov
Detlef Seese
Hans van Ditmarsch
Nikolas Vaporis
Thomas Vetterlein

Table of Contents

Applications of Finite Duality to Locally Finite Varieties of BL-Algebras

Stefano Aguzzoli[1], Simone Bova[2], and Vincenzo Marra[3]

[1] Università degli Studi di Milano, Dipartimento di Scienze dell'Informazione
via Comelico 39/41, I-20135 Milano, Italy
aguzzoli@dsi.unimi.it
[2] Università degli Studi di Siena, Dipartimento di Matematica e Informatica
Pian dei Mantellini 44, I-53100 Siena, Italy
bova@unisi.it
[3] Università degli Studi di Milano, Dipartimento di Informatica e Comunicazione
via Comelico 39/41, I-20135 Milano, Italy
marra@dico.unimi.it

Abstract. We are concerned with the subvariety of commutative, bounded, and integral residuated lattices, satisfying divisibility and prelinearity, namely, BL-algebras. We give an explicit combinatorial description of the category that is dual to finite BL-algebras. Building on this, we obtain detailed structural information on the locally finite subvarieties of BL-algebras that are analogous to Grigolia's subvarieties of finite-valued MV-algebras. As an illustration of the power of the finite duality presented here, we give an exact recursive formula for the cardinality of free finitely generated algebras in such varieties.

Keywords: BL-algebras, prime filters, dualities, free BL-algebras, subvarieties of BL-algebras, locally finite varieties.

1 Introduction

Hájek's Basic Logic BL [8] is the logic of all continuous triangular norms and their residua [4]. It is a fundamental object of study in the area of mathematical fuzzy logic, whose aim is to develop formal systems to make inferences in the presence of vagueness or uncertainty. The Lindenbaum-Tarski algebraic semantics of BL is given by the variety of BL-algebras, that is, commutative, bounded, integral residuated lattices satisfying divisibility and prelinearity.

To use such a tool as BL in practice, one needs to be able to manipulate BL-algebras effectively. In this direction, combinatorial representations of BL-algebras are of the foremost importance. In this paper we show that combinatorial representations are available for finite BL-algebras and locally finite[1] subvarieties of BL-algebras. Towards this aim, we shall introduce a full-fledged spectral duality for finite BL-algebras.

[1] A variety of algebras is locally finite if each finitely generated member of the variety is finite; equivalently, if finitely generated free algebras are finite.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 1–15, 2009.

It turns out that dual objects are finite *weighted* forests, that is, forests labeled with natural numbers. We define morphisms of weighted forests so as to provide a natural categorical equivalence with the opposite of the category of finite BL-algebras and their homomorphisms. Thus, any finite BL-algebra arises as the algebra of parts of a weighted forest, in an appropriate sense.

The combinatorial structure of forests allows us to effectively compute products and coproducts in the dual category, and then the duality affords a translation back to finite BL-algebras. This provides us with a powerful tool to extract structural information from finite BL-algebras. As an example of this machinery, we study the BL-algebraic analogous of Grigolia's subvarieties of finite-valued MV-algebras, obtaining the exact structure of their dual weighted forests, together with an exact recursive formula for the cardinality of free finitely generated algebras in such varieties.

2 Preliminaries

We write $\mathbb{N} = \{1, 2, \ldots\}$, $\setminus$ for set-theoretic difference, $a|b$ if $a, b \in \mathbb{N}$ and a divides b, and $|A|$ for the cardinality of the set A.

A *basic hoop* is an algebra $(A, \odot, \to, \wedge, \vee, \top)$ of type $(2, 2, 2, 2, 0)$ such that $(A, \odot, \top)$ is a commutative monoid, $(A, \wedge, \vee)$ is a lattice, and the following properties hold:

$$
\begin{array}{ll}
(\textit{residuation}) & x \odot y \leq z \text{ if and only if } x \leq y \to z, \\
(\textit{integrality}) & x \wedge \top = x, \\
(\textit{divisibility}) & x \wedge y = y \odot (y \to x), \\
(\textit{prelinearity}) & (x \to y) \vee (y \to x) = \top;
\end{array}
$$

equivalently, a basic hoop is a commutative, integral, divisible residuated lattice satisfying prelinearity. Note that basic hoops form a variety since residuation can be formulated by identities; see [8, 2.3.10]. A *Wajsberg hoop* is a basic hoop satisfying $\neg\neg x = x$. (Throughout, we use $\neg x$ as an abbreviation for $x \to \bot$.)

A *BL-algebra* is an algebra $(A, \odot, \to, \wedge, \vee, \bot, \top)$ of type $(2, 2, 2, 2, 0, 0)$ such that $(A, \odot, \to, \wedge, \vee, \top)$ is a *bounded* basic hoop, that is,

$$x \wedge \bot = \bot$$

holds. In each BL-algebra, the operations $\wedge$, $\vee$, and $\top$ are definable from the other operations, as follows: $x \wedge y = x \odot (x \to y)$, $x \vee y = ((x \to y) \to y) \wedge ((y \to x) \to x)$, and $\top = \bot \to \bot$; see [8, 2.1.10]. In the sequel, we shall therefore feel free to use the shorter signature $(A, \odot, \to, \bot)$ instead of the complete one, whenever convenient.

A BL-algebra is called: a *BL-chain*, if the reduct $(A, \wedge, \vee, \bot, \top)$ is totally ordered; a *Gödel algebra*, if it satisfies $x \odot x = x$ (elements of a BL-algebra satisfying $x \odot x = x$ are said to be *idempotent*); and an *MV-algebra* [3] if it satisfies $\neg\neg x = x$.

We shall make use of the *ordinal sum* construction. Let $\{(A_i, \odot_i, \to_i, \top_i)\}_{i \in I}$, for I a linearly ordered set with minimum 0, be a family of totally ordered

Wajsberg hoops such that $A_i \cap A_j = \{\top_i\} = \{\top_j\}$ for all $i, j \in I$, with A_0 bounded. Then the ordinal sum of the family $\{A_i\}_{i\in I}$ is the BL-algebra $(\bigcup_{i\in I} A_i, \odot, \to, \bot_0)$, where:

$$x \odot y = \begin{cases} x & \text{if } x \in A_i, y \in A_j, i < j \\ x \odot_i y & \text{if } x, y \in A_i \\ y & \text{if } x \in A_i, y \in A_j, j < i \end{cases}$$

$$x \to y = \begin{cases} \top_0 & \text{if } x \in A_i, y \in A_j, i < j \\ x \to_i y & \text{if } x, y \in A_i \\ y & \text{if } x \in A_i, y \in A_j, j < i \end{cases}$$

3 Spectral Duality for Finite BL-Algebras

Fix a finite BL-algebra A. Recall that a *filter* of A is a nonempty upper set[2] $F \subseteq A$ that is closed under $\odot$. Further, a filter F of A is *prime* if $x \to y \in F$ or $y \to x \in F$ for each $x, y \in F$. We write $\operatorname{Spec} A$ for the (*prime*) *spectrum* of A, the set of prime filters of A partially ordered by reverse inclusion. Congruences θ of A are in bijection with filters F of A via $F = \{x \in A \mid (x, \top) \in \theta\}$. Prime filters precisely correspond to those congruences θ on A such that A/θ is a BL-chain. See [8, 2.3.14] for details.

For any finite BL-chain C we define the *top part* of C to be

$$T(C) = \{x \in C \mid x > c,\ c \text{ the largest idempotent below } \top\} .$$

The *weighted spectrum* of A is the function $\operatorname{wSpec} A \colon \operatorname{Spec} A \to \mathbb{N}$ such that

$$\mathfrak{p} \mapsto |T(A/\mathfrak{p})| ,$$

for every prime filter $\mathfrak{p} \in \operatorname{Spec} A$.

Throughout, *poset* means *finite* partially ordered set (with the partial order relation usually denoted by $\leq$). A *forest* is a poset such that the collection of lower bounds of any given element is totally ordered. A *weighted forest* is a function $w \colon F \to \mathbb{N}$, where F is a forest. Consider two weighted forests $w \colon F \to \mathbb{N}$, $w' \colon F' \to \mathbb{N}$. By a *morphism* $g \colon w \to w'$ we mean an order-preserving map $g \colon F \to F'$ that is

(M1) *open* (or is a *p-morphism*), i.e. whenever $x' \leq g(x)$ for $x' \in F'$ and $x \in F$, then there is $y \leq x$ in F such that $g(y) = x'$, and
(M2) *respects weights*, meaning that for each $x \in F$, there exists $y \leq x$ in F such that $g(y) = g(x)$ and $w'(g(y))$ divides $w(y)$.

[2] A *lower set* of a poset P is a subset D such that $x \in D$ and $y \leq x \in P$ imply $y \in D$. The smallest lower set containing a subset $S \subseteq P$ is denoted $\downarrow S$. *Upper sets* and the notation $\uparrow S$ are defined analogously.

Contemplation of these definitions shows that weighted forests and their morphisms form a category. Let us write WF for the latter category, and FBL for the category of finite BL-algebras and their homomorphisms.

It is possible to prove that $\mathrm{wSpec}\, A$ is a weighted *forest* for any finite BL-algebra A. In fact, wSpec can be turned into a contravariant functor from FBL to WF, as follows. Given a homomorphism $h\colon A \to B$ of finite BL-algebras, one proves that there is a function

$$\mathrm{Spec}\, h\colon \mathrm{Spec}\, B \to \mathrm{Spec}\, A \tag{1}$$

defined by

$$\mathfrak{p} \in \mathrm{Spec}\, B \mapsto h^{-1}(\mathfrak{p}) \in \mathrm{Spec}\, A\ . \tag{2}$$

Moreover, one checks that $\mathrm{Spec}\, h$ is an open order-preserving map from the forest $\mathrm{Spec}\, B$ to the forest $\mathrm{Spec}\, A$, and that it respects the weights of $\mathrm{wSpec}\, B$ and $\mathrm{wSpec}\, A$. Hence, $\mathrm{Spec}\, h$ defines a morphism

$$\mathrm{wSpec}\, h\colon \mathrm{wSpec}\, B \to \mathrm{wSpec}\, A$$

of weighted forests. Direct inspection now shows that wSpec sends identity maps to identity maps, and preserves composition. To sum up, wSpec is a contravariant functor from FBL to WF.

Conversely, we next construct a contravariant functor from weighted forests to finite BL-algebras. If F is a forest, a *subforest* of F is any lower set of F. If $w\colon F \to \mathbb{N}$ is a weighted forest, a *weighted subforest* of w is defined as any $w'\colon F' \to \mathbb{N}$ with F' a subforest of F such that $w'(x) \leq w(x)$ for all $x \in \max F'$, and $w'(x) = w(x)$ otherwise. We write $\mathrm{Sub}\, w$ for the set of all weighted subforests of w.

It turns out that $\mathrm{Sub}\, w$ carries a natural structure of BL-algebra, as follows. To begin with, writing $\emptyset\colon \emptyset \to \mathbb{N}$ for the unique empty weighted forest, we set $\bot = \emptyset$, and $\top = w$. To define $\odot$, consider subforests $u\colon U \to \mathbb{N}$ and $v\colon V \to \mathbb{N}$ of w. Define a function $a\colon U \cap V \to \mathbb{N} \cup \{0\}$ by

$$a(x) = \begin{cases} \max(0, u(x) + v(x) - w(x)) & \text{if } x \in \max U \cap \max V \\ u(x) & \text{if } x \in \max U \text{ and } x \notin \max V \\ v(x) & \text{if } x \notin \max U \text{ and } x \in \max V \\ w(x) & \text{otherwise,} \end{cases}$$

for each $x \in U \cap V$. Let $E = \{x \in U \cap V \mid a(x) > 0\}$, and define $u \odot v\colon E \to \mathbb{N}$ by the restriction $u \odot v = a \restriction E$.

Turning to implication, we define $u \to v$. First, we set

$$\begin{aligned} A &= F \setminus {\uparrow}(U \setminus V)\ , \\ B &= \{x \mid x \in \max U \cap \max V \text{ and } u(x) > v(x)\}\ . \end{aligned}$$

Then we set $E = (A \setminus \uparrow B) \cup B$. We define $(u \to v)\colon E \to \mathbb{N}$ by

$$(u \to v)(x) = \begin{cases} v(x) + w(x) - u(x) & \text{if } x \in B \\ v(x) & \text{if } x \in (U \cap V) \setminus (\max U \cap \max V) \\ w(x) - u(x) & \text{if } x \in \min U \setminus V \\ w(x) & \text{otherwise,} \end{cases}$$

for each $x \in E$.

Lattice operations $u \vee v\colon U \cup V \to \mathbb{N}$ and $u \wedge v\colon U \cap V \to \mathbb{N}$ turn out to be as follows:

$$(u \vee v)(x) = \begin{cases} \max(u(x), v(x)) & \text{if } x \in U \text{ and } x \in V \\ u(x) & \text{if } x \in U \text{ and } x \notin V \\ v(x) & \text{if } x \notin U \text{ and } x \in V \end{cases}$$

for each $x \in U \cup V$, and

$$(u \wedge v)(x) = \begin{cases} \min(u(x), v(x)) & \text{if } x \in \max U \cap \max V \\ u(x) & \text{if } x \in \max U \text{ and } x \notin \max V \\ v(x) & \text{if } x \notin \max U \text{ and } x \in \max V \\ w(x) & \text{otherwise,} \end{cases}$$

for each $x \in U \cap V$, respectively.

It can now be proved that for any weighted forest $w\colon F \to \mathbb{N}$, the algebraic structure

$$(\operatorname{Sub} w, \odot, \to, \wedge, \vee, \bot, \top)$$

is a (finite) BL-algebra. To turn Sub into a contravariant functor from WF to FBL, we take inverse images again. Namely, if $g\colon w \to w'$ is a morphism between the weighted forests $w\colon F \to \mathbb{N}$ and $w'\colon F' \to \mathbb{N}$, we define $\operatorname{Sub} g\colon \operatorname{Sub} F' \to \operatorname{Sub} F$ by

$$U \in \operatorname{Sub} F' \mapsto g^{-1}(U) \in \operatorname{Sub} F \ .$$

One can prove that $\operatorname{Sub} g$ so defined is a homomorphism of BL-algebras. To sum up, Sub is a contravariant functor from WF to FBL.

Finally, one can prove that wSpec and Sub yield a duality. Here we omit the proof for space constraints.

Theorem 1 (Finite Duality). *The category of finite BL-algebras and their homomorphisms is dually equivalent to the category of weighted forests and their morphisms. That is, the composite functors* wSpec $\circ$ Sub *and* Sub $\circ$ wSpec *are naturally isomorphic to the identity functors on* WF *and* FBL*, respectively.*

In particular, by [10, Thm. IV.4.1] the functor wSpec is essentially surjective, and this yields the following representation theorem for finite BL-algebras.

Corollary 1. *Any finite BL-algebra is isomorphic to* $(\operatorname{Sub} w, \odot, \to, \wedge, \vee, \bot, \top)$*, for a weighted forest* $w\colon F \to \mathbb{N}$ *that is unique to within an isomorphism of weighted forests.*

While the previous corollary has already been proved in [6, §5] and, as a special case of a more general construction, in [9, §6], the finite duality theorem is a novelty. In the rest of the paper, we illustrate the potential of this duality for the investigation, and possibly the classification, of locally finite subvarieties of BL-algebras.

4 Grigolia's Subvarieties of BL-Algebras

In the variety of BL-algebras, we adopt the abbreviation $x \oplus y$ for the binary term operation

$$((x \to (x \odot y)) \to y) \vee ((y \to (y \odot x)) \to x)\,.$$

In each MV-algebra, one has $((x \to (x \odot y)) \to y) \vee ((y \to (y \odot x)) \to x) = \neg(\neg x \odot \neg y)$, that is, $\oplus$ coincides with the Łukasiewicz sum. Thus, our usage of $\oplus$ is consistent with standard MV-algebraic notation. Further, it is an exercise to check that, in every BL-algebra, the operation $\oplus$ is commutative, associative, and satisfies $x \oplus \top = \top$; cf. [2, Definition 2.2]. Thus, we can consistently shorten $x \oplus x \oplus \cdots \oplus x$ to hx, and similarly $x \odot x \odot \cdots \odot x$ to x^h, where in both cases x occurs h many times, for $h > 0$ an integer. Finally, we set $x^0 = \top$ and $0x = \bot$.

In [7, pag. 81–82], Grigolia axiomatized the variety $\mathbb{MV}_k$ generated by the k-element MV-chain $\mathbb{L}_k$, for each integer $k \geq 2$, extending the axioms for MV-algebras by the following axiom schemata.[3]

(G1) $x^k = x^{k-1}$,
(Gh) $k(x^h) = (h(x^{h-1}))^k$, for every integer $2 \leq h \leq k-2$ that does not divide $k-1$.

For a given $k \geq 2$, we define $\mathbb{BL}_k$ to be the variety of BL-algebras satisfying (G1–Gh), for all integers h such that $2 \leq h \leq k-2$, and such that h is not a divisor of $k-1$.

Note that $\mathbb{BL}_k$ contains the variety of Gödel algebras. Indeed, one checks that each Gödel algebra satisfies $x^h = x^k$ and $hx = kx$ for every $h, k > 0$, so that axioms (G1) and (Gh) boil down to $x = x$.

For $k, l \in \mathbb{N}$, we write B_k^l to denote the ordinal sum of l copies of $\mathbb{L}_k$.

Lemma 1. *Fix $k \geq 2$.*

(1) *The variety $\mathbb{BL}_k$ is generated by $\{B_k^l \mid l \in \mathbb{N}\}$.*
(2) *The variety $\mathbb{BL}_k$ is locally finite.*
(3) *For a finite BL-algebra A, the following are equivalent.*
  (i) $A \in \mathbb{BL}_k$.
  (ii) $\mathrm{wSpec}\, A$ *has range included in the set of divisors of* $k-1$.

[3] Here, Grigolia's axioms are presented in the version adopted in [3, 8.5.1].

Proof. (1) Suppose a term $\tau(X_1, \ldots, X_n)$ in the language of BL-algebras fails — i.e., evaluates to an element $\neq \top$ — in a BL-algebra B lying in $\mathbb{BL}_k$. Since each BL-algebra is a subdirect product of BL-chains ([8, 2.3.16]), it is safe to assume that B is a chain. By [1, Theorem 3.7], B is an ordinal sum of totally ordered Wajsberg hoops, the first of which is bounded (equivalently, is an MV-chain). Moreover, τ fails in a finitely generated subalgebra of B. Since a finitely generated subalgebra of an ordinal sum of Wajsberg hoops is an ordinal sum of finitely many components, we may assume that B is a finite ordinal sum of Wajsberg hoops W_i, $i = 1, \ldots, l$, with W_1 an MV-chain. Either by direct inspection, or by the argument in [11, Theorem 1], one sees it is safe to assume $l = n + 1$. Since B satisfies (G1), if $x \in W_i \setminus \{\top\}$ then x^{k-1} is idempotent. But then x^{k-1} must be the bottom of W_i. Indeed, it is well-known that the only idempotents of linearly ordered Wajsberg hoops are the top and (when it exists) the bottom element. Therefore, each W_i is an MV-chain. Since by hypothesis this MV-chain satisfies (Gh) and (Gh) is $\bot$-free, $h \geq 1$, it follows that each W_i lies in $\mathbb{MV}_k$. In conclusion, since $\mathbb{MV}_k$ is generated by $\mathrm{Ł}_k$, τ fails in the ordinal sum of $n + 1$ copies of $\mathrm{Ł}_k$, as was to be shown.

(2) This is an immediate consequence of the fact that $\mathbb{MV}_k$ is locally finite by [3, 8.6.1], along with the observation that an n-generated BL-chain is the ordinal sum of at most $n + 1$ summands.

(3) (i) $\Rightarrow$ (ii). If $\mathfrak{p}$ is a prime filter of A, the top part $T(A/\mathfrak{p})$ of the BL-chain $A/\mathfrak{p}$ can be made into an MV-chain lying in $\mathbb{MV}_k$ by adding a bottom element to it, and extending the operations in the only possible way. If such an MV-chain has cardinality c, then $c - 1$ divides $k - 1$, hence (ii) follows.
(ii) $\Rightarrow$ (i). Suppose $A \notin \mathbb{BL}_k$. Then there is a prime filter $\mathfrak{p}$ of A such that the BL-chain $A/\mathfrak{p}$ does not lie in $\mathbb{BL}_k$. Equivalently, $A/\mathfrak{p}$ is an ordinal sum of finitely many finite MV-chains $\mathrm{Ł}_{c_i}$, $i = 1, \ldots, u$, and there exists $j \in \{1, \ldots, u\}$ such that $\mathrm{Ł}_{c_j}$ does not lie in $\mathbb{MV}_k$. The latter condition means that $c_j - 1$ does not divide $k - 1$. Let $\mathfrak{q}$ be the prime filter of $A/\mathfrak{p}$ generated by the bottom of $\mathrm{Ł}_{c_{j+1}}$, if $j < u$; otherwise, let $\mathfrak{q}$ be the trivial filter $\{\top\}$ of $A/\mathfrak{p}$. Now $|T((A/\mathfrak{p})/\mathfrak{q})|$ does not divide $k - 1$ by construction, and $|T((A/\mathfrak{p})/\mathfrak{q})|$ is in the range of $\mathrm{wSpec}\, A$ by the isomorphism theorems.

5 The Weighted Spectrum of Free Algebras in $\mathbb{BL}_k$

We write $\mathrm{Free}_{n,k}$ for the free n-generated algebra in $\mathbb{BL}_k$, for $k \geq 2$ and $n \geq 0$ an integer. By [3, 8.6.1], the free n-generated MV-algebra in $\mathbb{MV}_k$ is given by the direct product

$$\mathrm{FreeMV}_{n,k} = \prod_{d \,|\, (k-1)} \mathrm{Ł}_{d+1}^{\alpha(n,d)} \,, \tag{3}$$

where $d \in \mathbb{N}$, $\alpha(0, d)$ is 0 if $d > 1$ and 1 if $d = 1$, and, for $n \geq 1$,

$$\alpha(n, d) = (d + 1)^n + \sum_{\emptyset \neq X \subseteq \mathrm{PrDiv}\,(d)} (-1)^{|X|} (\gcd X + 1)^n \,, \tag{4}$$

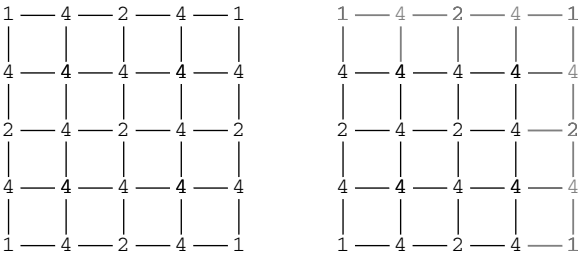


Fig. 1. $\alpha(1,4) = 4$, $\alpha(2,4) = 5$, $\alpha(4,4) = 16$ (left), and $\beta(1,4) = 1$, $\beta(2,4) = 3$, $\beta(4,4) = 12$ (right)

where $\mathrm{PrDiv}\,(d)$ is the set of coatoms in the lattice of divisors of d. Geometrically, $\alpha(n,d)$ counts the number of points in $[0,1]^n$ whose denominator is d.[4] We now define a variant of α. We let $\beta(0,d) = 0$ if $d > 1$ and 1 if $d = 1$, and, for $n \geq 1$,

$$\beta(n,d) = d^n + \sum_{\emptyset \neq X \subseteq \mathrm{PrDiv}\,(d)} (-1)^{|X|} (\gcd X)^n \,. \tag{5}$$

Geometrically, $\beta(n,d)$ counts the number of points in $[0,1)^n$ whose denominator is d — in other words, β does not take into account those points of $[0,1]^n$ having at least one coordinate set to 1. Compare Figure 1 for an example. Notice that letting $L(d) = \{\, (e_1,e_2) \mid \mathrm{lcm}\,(e_1,e_2) = d \,\}$, where $\mathrm{lcm}\,(a,b)$ denotes the least common multiple of integers a and b, we have

$$\sum_{(e_1,e_2)\in L(d)} \beta(h,e_1)\beta(k,e_2) = \beta(h+k,d)\,. \tag{6}$$

We next use (5) to define a family of weighted forests that shall be proved dual to free algebras in $\mathbb{BL}_k$. To this purpose, we introduce some additional notation. If U is a forest we write $U_\perp$ for the forest obtained from U by adding a new bottom element $\perp$. Further, if $u\colon U \to \mathbb{N}$ is a weighted forest, we write $u_\perp$ for the weighted forest having $U_\perp$ as domain, and such that $u_\perp$ agrees with u over U, and $u_\perp(\perp) = 1$. It is a standard fact that varieties of algebras are both complete and cocomplete. If, moreover, $\mathbb{V}$ is a subvariety of the variety $\mathbb{W}$, then a product of $\mathbb{V}$-algebras computed in $\mathbb{V}$ coincides with the same product computed in $\mathbb{W}$. Then Theorem 1 implies at once that WF has all finite coproducts. We write $u+v$ for the coproduct of the weighted forests $u\colon U \to \mathbb{N}$ and $v\colon V \to \mathbb{N}$. Clearly, since products in varieties of algebras are Cartesian, we have $u+v\colon U+V \to \mathbb{N}$, where $U+V$ is the disjoint union of U and V, and $u+v$ agrees with u on U, and with v on V. Thus, up to an isomorphism, any weighted forest can be written as a coproduct $\sum_{i=1}^{u} T_i$ of weighted trees in essentially just one way. Here, as usual, a *tree* is a forest with a minimum element.

[4] The *denominator* of a rational point $(r_1/s_1,\ldots,r_n/s_n)$, where $r_i, s_i \geq 0$ are integers such that $s_i \neq 0$ and r_i and s_i are relatively prime, is the least common multiple of the set s_i's.

For coproducts in varieties and subvarieties the situation is generally not as simple. In our case, however, we have the following properties (proofs of the following two lemmas are omitted for space constraints).

Lemma 2. (i) *The category* WF *has all finite products.* (ii) *If* u, v *are weighted forests, and* $u \times v$ *is their product with projections* $\pi_u \colon u \times v \to u$ *and* $\pi_v \colon u \times v \to v$*, then* $\operatorname{Sub} u \times v$*, along with the injections* $\operatorname{Sub} \pi_u$*,* $\operatorname{Sub} \pi_v$*, is the coproduct of* $\operatorname{Sub} u$ *and* $\operatorname{Sub} v$ *in the category of all BL-algebras.* (iii) *For each integer* $k \geq 2$*, coproducts computed in* $\mathbb{BL}_k$ *coincide with coproducts computed in the category of all BL-algebras.* (iv) *For any three weighted forests* u, v, w*, we have* $u \times (v + w) \cong (u \times v) + (u \times w)$.

Note that, using (iv) in Lemma 2, we obtain the binomial expansion

$$(u + v)^m \cong \sum_{i=0}^{m} \binom{m}{i} u^i v^{m-i} \ ,$$

for any two weighted forests u, v. Here and in the sequel, in the expressions involving products and coproducts we adopt the standard notation of elementary arithmetic.

We now describe the finite products in the category WF. Let $v \colon F_v \to \mathbb{N}$ and $w \colon F_w \to \mathbb{N}$ be two weighted forests. By $F_v \times F_w$ we mean the product of the underlying forests as described in [5]. We further denote by $\pi_v \colon (F_v \times F_w) \to F_v$ and $\pi_w \colon (F_v \times F_w) \to F_w$ the associated projections. The product of v and w in WF is the function $(v \times w) \colon (F_v \times F_w) \to \mathbb{N}$ together with the projections π_v and π_w, defined as follows.

Pick $p \in F_v$ and $q \in F_w$. In case $p \notin \min F_v$ and $q \notin \min F_w$ there are exactly three disjoint classes of points in $F_v \times F_w$, denoted by $(p|q)$, (p, q), $(q|p)$, such that all points in them project through π_v to p and through π_w to q. Further, any point in $F_v \times F_w$ that projects to p and q, respectively, falls into one of these classes. In particular, by [5], each point in $(p|q)$ is such that its predecessor in $F_v \times F_w$ projects to p through π_v and to the predecessor of q in F_w through π_w. The case of the class of points $(q|p)$ is symmetric. Each point in (p, q) is such that its predecessor in $F_v \times F_w$ projects to the respective predecessors through both projections.

If exactly one point in $\{p, q\}$ is minimal in the forest it belongs, say $p \in \min F_v$, then there is exactly one point in $F_v \times F_w$, precisely in $(p|q)$, such that its predecessor projects to p through π_v and to the predecessor of q in F_w through π_w. If $p \in \min F_v$ and $q \in \min F_w$ then there is exactly one point in $F_v \times F_w$, classed in (p, q), such that $\pi_v(p, q) = p$ and $\pi_w(p, q) = q$. Moreover, $(p, q) \in \min(F_v \times F_w)$.

Let $r \in F_v \times F_w$ be such that $\pi_v(r) = p$ and $\pi_w(r) = q$. Then

$$(v \times w)(r) = \begin{cases} w(q) & \text{if } r \in (p|q) \\ v(p) & \text{if } r \in (q|p) \\ \operatorname{lcm}(v(p), w(q))) & \text{if } r \in (p, q) \end{cases}$$

where $\mathrm{lcm}\,(a,b)$ denotes the least common multiple of $a, b \in \mathbb{N}$.

Throughout, let P_d denote the weighted tree consisting of just one point having weight d, for $d \in \mathbb{N}$. The description of finite products in WF given above allows to prove the following properties.

Lemma 3. *Let $u_\perp$ and $v_\perp$ be two trees in WF. If $u \neq \emptyset \neq v$ then*

$$u_\perp \times v_\perp \cong (u \times v_\perp + u \times v + u_\perp \times v)_\perp\,.$$

Further, $P_d P_e \cong P_{\mathrm{lcm}\,(d,e)}$, $P_d u_\perp \times P_e v_\perp \cong P_{\mathrm{lcm}\,(d,e)}(u_\perp \times v_\perp)$, $P_1 u_\perp \cong u_\perp$.

For $k \geq 2$, we introduce the following definitions:

- $M_k^0 = P_1$.
- For each integer $n \geq 1$,

$$M_k^n = \sum_{d|(k-1)} \beta(n,d) P_d\ .$$

Note in particular that if $k-1$ is prime, or equal to 1, then $M_k^1 = P_1 + (k-2)P_{k-1}$.

Lemma 4. *Fix $k \geq 2$. Set $F_k^1 = M_k^1 + (M_k^1)_\perp$. Then*

$$\mathrm{wSpec}\,\mathrm{Free}_{1,k} \cong F_k^1\ .$$

Proof. If S is any set and B is a BL-algebra, let us write B^S for the BL-algebra of all functions $S \to B$ endowed with the operations inherited pointwise from B. By the argument proving (1) in Lemma 1, we know that if a term $\tau(X_1)$ in the language of BL-algebras fails in some BL-algebra lying in $\mathbb{BL}_k$, then it must fail in B_k^2. Hence, by standard universal-algebraic considerations, $\mathrm{Free}_{1,k}$ is (isomorphic to) the subalgebra of $(B_k^2)^{B_k^2}$ that is generated by the identity function, the latter being a free generator. Let us write $C \cong \mathrm{Ł}_k$ and $D \cong \mathrm{Ł}_k$ for the first and second summand of B_k^2, respectively, and b for the bottom element of D (i.e., the unique idempotent of B_k^2 besides top and bottom). A trivial structural induction shows that any element $f \in \mathrm{Free}_{1,k}$ is such that (i) $f(p/q) = r/q$ for p/q an irreducible fraction in $[0,2]$ such that q divides $k-1$; (ii) $f(c) \in C$ for any $c \in C$; (iii) $f(\top) = \perp$ implies $f(d) = \perp$, while $f(\top) = \top$ implies $f(d) \geq b$, for any $d \in D$. Conversely, let $f \in (B_k^2)^{B_k^2}$ satisfy (i–iii) above. A straightforward adaptation of [11, Thm. 2] shows that $f \in \mathrm{Free}_{1,k}$. As an immediate consequence of this representation of $\mathrm{Free}_{1,k}$ it follows that $\mathrm{Spec}\,\mathrm{Free}_{1,k}$ is isomorphic to the underlying forest of F_k^1. A further computation confirms that $\mathrm{wSpec}\,\mathrm{Free}_{1,k}$ is isomorphic to F_k^1. As a matter of fact, each prime filter of $\mathrm{Free}_{1,k}$ is singly generated by a function $f \in \mathrm{Free}_{1,k}$ of one of the following three types: (1) there exists $c \in C \setminus \{\top\}$ such that $f(c) = \top$ and $f(a) = 0$ for all $c \neq a \in B_k^2$; (2) there exists $d \in D \setminus \{\top\}$ such that $f(d) = f(\top) = \top$, while $f(e) = b$ for all $e \in D \setminus \{d, \top\}$, and $f(c) = 0$ for all $c \in C \setminus \{\top\}$; (3) $f(\top) = \top$, $f(d) = b$ for all $d \in D \setminus \{\top\}$, and $f(c) = 0$ for all $c \in C \setminus \{\top\}$. Notice that the only filter of type (3) includes all filters of type (2) and no other inclusions hold in $\mathrm{wSpec}\,\mathrm{Free}_{1,k}$.

Fig. 2. F_3^1 (left) and F_7^1 (right)

Example 1. Figure 2 displays the weighted forest F_3^1 (to the left), and the weighted forest F_7^1 (to the right).

For each $k \geq 2$ and each integer $n \geq 1$, let us define

$$F_k^n = \underbrace{F_k^1 \times \cdots \times F_k^1}_{n \text{ times}} = (F_k^1)^n \ .$$

Lemma 5. *Fix $k \geq 2$, and for each integer $n \geq 0$,*

$$F_k^n \cong \mathrm{wSpec}\,\mathrm{Free}_{n,k} \ .$$

Proof. By standard universal algebra, in any variety the free algebra on κ free generators, for κ a cardinal, is isomorphic to the copower of κ-many copies of the free algebra on one generator. Thus, $\mathrm{Free}_{n,k} \cong \sum_{i=1}^n \mathrm{Free}_{1,k}$, where the right-hand side coproduct is computed in $\mathbb{BL}_k$. Using Lemma 2 we have $\mathrm{wSpec}\,\mathrm{Free}_{n,k} \cong \mathrm{wSpec}\,(\sum_{i=1}^n \mathrm{Free}_{1,k}) \cong \prod_{i=1}^k \mathrm{wSpec}\,\mathrm{Free}_{1,k}$. By Lemma 4, $\prod_{i=1}^k \mathrm{wSpec}\,\mathrm{Free}_{1,k} \cong \prod_{i=1}^k F_k^1 = F_k^n$, and the lemma is proved.

Our next objective is to obtain an explicit description of F_k^n for any n and k. To this aim, we define the following weighted trees. Fix $k \geq 2$, and an integer $d \geq 1$:

- $T_{k,d}^0 = P_d$.
- For each integer $n \geq 1$,

$$T_{k,d}^n = P_d \left(\sum_{i=1}^n \sum_{e|k-1} \binom{n}{i} \beta(i,e) T_{k,e}^{n-i} \right)_\perp \ .$$

Lemma 6. *Fix integers $k \geq 2$, $d \geq 1$, and $m \geq 1$.*

(1) $T_{k,1}^1 \cong (M_k^1)_\perp$.
(2) $T_{k,d}^m \cong P_d T_{k,1}^m$.
(3) $M_k^m \cong (M_k^1)^m$.
(4) $T_{k,d}^m \cong (T_{k,d}^1)^m$.

Proof. (1) follows immediately from the definition of $T_{k,1}^1$, $T_{k,1}^0$ and M_k^1. (2) follows from Lemma 3 and the definition of $T_{k,1}^1$ and $T_{k,1}^m$.

(3) By induction on m. The base case is trivial. Write $(M_k^1)^m$ as $M_k^1 \times (M_k^1)^{m-1}$. By induction, $M_k^{m-1} \cong (M_k^1)^{m-1}$. By Lemma 3, since products distribute over coproducts,

$$\begin{aligned} M_k^1 \times M_k^{m-1} &\cong \sum_{d|k-1} \beta(1,d) P_d \times \sum_{e|k-1} \beta(m-1,e) P_e \\ &\cong \sum_{d|k-1} \sum_{e|k-1} \beta(1,d)\beta(m-1,e) P_{\mathrm{lcm}\,(d,e)} \\ &\cong \sum_{d|k-1} \beta(m,d) P_d \cong M_k^m \qquad \text{(by (6))}. \end{aligned}$$

(4) By induction on m. The base case is trivial. Write $(T_{k,d}^1)^m$ as $T_{k,d}^1 \times (T_{k,d}^1)^{m-1}$. By induction $T_{k,d}^{m-1} \cong (T_{k,d}^1)^{m-1}$. Let

$$V = \sum_{i=1}^{m-1} \sum_{e|k-1} \binom{m-1}{i} \beta(i,e) T_{k,e}^{m-1-i},$$

so $P_d V_\perp \cong T_{k,d}^{m-1}$. By Lemma 3,

$$T_{k,d}^1 \times T_{k,d}^{m-1} \cong P_d \left((M_k^1)_\perp \times V_\perp \right) \cong P_d \left((M_k^1 \times V_\perp) + (M_k^1 \times V) + (T_{k,1}^1 \times V) \right)_\perp .$$

By distributivity and Lemma 3, since M_k^1 is a forest of one-point trees:

$$M_k^1 \times V_\perp \cong \sum_{e|k-1} \beta(1,e) P_e V_\perp \cong \sum_{e|k-1} \beta(1,e) T_{k,e}^{m-1};$$

analogously,

$$\begin{aligned} M_k^1 \times V &\cong \left(\sum_{e|k-1} \beta(1,e) P_e \right) \times \left(\sum_{i=1}^{m-1} \sum_{e|k-1} \binom{m-1}{i} \beta(i,e) T_{k,e}^{m-1-i} \right) \\ &\cong \sum_{i=1}^{m-1} \sum_{e_1|k-1} \sum_{e_2|k-1} \binom{m-1}{i} \beta(1,e_1)\beta(i,e_2) T_{k,\mathrm{lcm}\,(e_1,e_2)}^{m-(i+1)} \\ &\cong \sum_{i=2}^{m-1} \sum_{e|k-1} \binom{m-1}{i-1} \beta(i,e) T_{k,e}^{m-i} \qquad \text{(by (6))}; \end{aligned}$$

finally,

$$\begin{aligned} T_{k,1}^1 \times V &\cong \sum_{i=1}^{m-1} \sum_{e|k-1} \binom{m-1}{i} \beta(i,e) (T_{k,1}^1 \times T_{k,e}^{m-1-i}) \\ &\cong \sum_{i=1}^{m-1} \sum_{e|k-1} \binom{m-1}{i} \beta(i,e) T_{k,e}^{m-i} \qquad \text{(Induction Hypothesis)} \\ &\cong \sum_{i=2}^{m-1} \sum_{e|k-1} \binom{m-1}{i} \beta(i,e) T_{k,e}^{m-i} + \sum_{e|k-1} (m-1)\beta(1,e) T_{k,e}^{m-1}. \end{aligned}$$

Summing up we have

$$T^1_{k,d} \times T^{m-1}_{k,d} \cong P_d\left((M^1_k \times V_\perp) + (M^1_k \times V) + (T^1_{k,1} \times V)\right)_\perp$$
$$\cong P_d\left(\sum_{i=1}^{m} \sum_{e|k-1} \binom{m}{i} \beta(i,e) T^{m-i}_{k,e}\right)_\perp ,$$

as was to be proved.

Example 2. The rightmost trees in Figure 2 are $T^1_{3,1}$ (left) and $T^1_{7,1}$ (right). The rightmost tree in Figure 3 is $T^2_{3,1}$.

We are finally in a position to exhibit the promised explicit description of $\mathrm{wSpec}\,\mathrm{Free}_{n,k}$.

Theorem 2. *For each $k \geq 2$ and each integer $n \geq 0$,*

$$F^n_k \cong \sum_{i=0}^{n} \sum_{d|k-1} \beta(i,d) \binom{n}{i} T^{n-i}_{k,d} \ .$$

Proof. By definition,

$$F^n_k = (F^1_k)^n = (M^1_k + (M^1_k)_\perp)^n \cong (M^1_k + T^1_{k,1})^n \ . \tag{7}$$

Since products distribute over coproducts, (7) yields

$$(M^1_k + T^1_{k,1})^n \cong \sum_{i=0}^{n} \binom{n}{i} (M^1_k)^i (T^1_{k,1})^{n-i} \ . \tag{8}$$

By Lemma 6 along with the definition of M^i_k, from (8) we deduce

$$(M^1_k + T^1_{k,1})^n \cong \sum_{i=0}^{n} \binom{n}{i} \sum_{d|k-1} \beta(i,d) P_d T^{n-i}_{k,1} \ . \tag{9}$$

As by Lemma 6.(2), $P_d T^{n-i}_{k,1} \cong T^{n-i}_{k,d}$, the lemma follows from (9) at once.

Example 3. Figure 3 displays the weighted forest F^2_3.

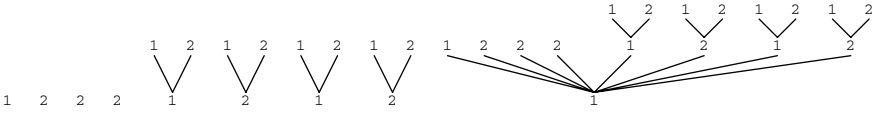


Fig. 3. $F^2_3 = (F^1_3)^2$

6 The Cardinality of Free Algebras in $\mathbb{BL}_k$

In this final section, we use Theorems 1 and 2 to obtain the cardinality of $\mathrm{Free}_{n,k}$. We shall write $t(k,n,d)$ for the cardinality of the BL-algebra $\mathrm{Sub}\, T^n_{k,d}$, where $k \geq 2$, $d \in \mathbb{N}$, and $n \geq 0$ an integer.

Lemma 7. *Fix $k \geq 2$, and an integer $n \geq 0$. Then $t(k,0,d) = d+1$, and*

$$t(k,n,d) = d + \prod_{i=1}^{n} \prod_{e|k-1} t(k,n-i,e)^{\binom{n}{i}\beta(i,e)} \ .$$

Proof. Follows immediately from Lemma 6.(4).

Theorem 3. *Fix $k \geq 2$, and an integer $n \geq 0$. Then:*

$$|\mathrm{Free}_{n,k}| = \prod_{i=0}^{n} \prod_{d|k-1} t(k,n-i,d)^{\binom{n}{i}\beta(i,d)} \ .$$

Proof. Follows immediately from Theorem 2.

To conclude, in the following table we report the cardinalities of $\mathrm{Free}_{n,k}$ for some values of n and k, computed using Theorem 3. Approximations are from below.

	$n=1$	$n=2$	$n=3$
$k=2$	6	342	137186159382
$k=3$	42	28677559680	$\sim 2.255534588 \cdot 10^{91}$
$k=4$	1056	$\sim 4.587963634 \cdot 10^{28}$	$\sim 1.230577614 \cdot 10^{373}$
$k=5$	22650	$\sim 1.525862962 \cdot 10^{55}$	$\sim 4.141165490 \cdot 10^{957}$
$k=6$	6721056	$\sim 1.738126059 \cdot 10^{106}$	$\sim 2.246803010 \cdot 10^{2299}$

References

1. Aglianó, P., Montagna, F.: Varieties of BL-algebras I: General Properties. J. Pure Appl. Algebra 181, 105–129 (2003)
2. Aguzzoli, S., Bova, S.: The Free n-Generated BL-algebra (submitted)
3. Cignoli, R.L.O., D'Ottaviano, I.M.L., Mundici, D.: Algebraic Foundations of Many-Valued Reasoning. Kluwer, Dordrecht (2000)
4. Cignoli, R.L.O., Esteva, F., Godo, L., Torrens, A.: Basic Fuzzy Logic is the Logic of Continuous t-norms and their Residua. Soft Comput. 4, 106–112 (2000)
5. D'Antona, O.M., Marra, V.: Computing Coproducts of Finitely Presented Gödel Algebras. Ann. Pure Appl. Logic 142, 202–211 (2006)
6. Di Nola, A., Lettieri, A.: Finite BL-algebras. Discrete Math. 269, 93–122 (2003)
7. Grigolia, R.S.: Algebraic Analysis of Łukasiewicz-Tarski's n-valued Logical Systems. In: Wójcicki, R., Malinowski, G. (eds.) Łukasiewicz Sentential Calculi, pp. 81–92. Ossolineum, Wrokław (1977)
8. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)

9. Jipsen, P., Montagna, F.: The Blok-Ferreirim Theorem for Normal GBL-algebras and its Application. Algebra Universalis (to appear)
10. MacLane, S.: Categories for the Working Mathematician, 2nd edn. Springer, New York (1998)
11. Montagna, F.: The Free BL-algebra on One Generator. Neural Network World 5, 837–844 (2000)

Completeness Results for Memory Logics

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

[1] INRIA Nancy Grand Est, France
[2] Departamento de Computación, FCEyN, UBA, Argentina
[3] CONICET, Argentina

Abstract. Memory logics are a family of modal logics in which standard relational structures are augmented with data structures and additional operations to modify and query these structures. In this paper we present sound and complete axiomatizations for some members of this family. We analyze the use of nominals to achieve completeness, and present one example in which they can be avoided.

1 Modal Logics with Memory

Many attempts have been made in recent years to increase modal logic expressivity by adding some notion of *state* to standard relational structures. This is a natural need, since modal logics are used in many different scenarios as tools for modeling behavior.

One example of how this can be achieved comes from epistemic logic with dynamic operators, which allow to express the evolution of knowledge by knowledge-changing actions. Such logics are often called Dynamic Epistemic Logics (DEL) [16], and a large number of DELs has been proposed [9,13,14,15]. These logics differ considerably in expressive power, but the common idea is to express knowledge evolution by accessing and changing the model structure through logic operators. An alternative approach comes from the software verification community, and the use of temporal logic with explicit global clocks which are accessed and controlled through logic operators. The logic XCTL of Harel et al. [10] is an example of this approach. Another example, also from the software verification community, is the extension of temporal logic with a concrete domain (e.g., the natural numbers with some operations like addition, comparison, etc.) which is accessed via the so-called freeze operator [1,11]. In the extended language, we can model qualitative properties using the temporal operators, and concrete properties –such as weight, temperature, etc.– using the new machinery. To cite yet another example, concrete domains have also been added to description logics, with much the same aims [12].

We would like to take a step back, and analyze some of the basic intuitions that most of the formal languages mentioned above have in common. We want to try to investigate the idea of adding an *explicit state* to a model, and being able to access (and modify) it via logical operators. And we would like to take this idea in its simplest form, in order to be able to understand it in detail.

⋆ S. Mera is partially supported by a grant of Fundación YPF.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 16–30, 2009.

We can take a standard relational structure and complement it with a *data structure*, that will keep the state information we want to model. We will also add to the logical language a collection of operations to modify and access the data structure. Formally, given a relational structure $\langle D, (R_r)_{r\in\mathsf{Rel}}, L\rangle$ where D is a non empty domain, $(R_r)_{r\in\mathsf{Rel}}$ is a set of relations over D, and $L : \mathsf{Atom} \to 2^D$ is a labeling function that assigns atomic properties to elements of D, we extend the structure with a set $S \subseteq D$. We can think of S as a set of states that are 'known' to us, and it will represent our current 'memory'. Even in this simple setting we can define the following operators:

$$\begin{array}{l}\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle, w \models \textcircled{r}\varphi \;\text{ iff }\; \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\cup\{w\}\rangle, w \models \varphi \\ \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle, w \models \textcircled{k} \;\text{ iff }\; w \in S.\end{array}$$

As it is clear from the definition above, the 'remember' operator ⓡ (a unary modality) just marks the current state as being 'known' or 'already visited', by storing it in our 'memory' S. On the other hand, the zero-ary operator ⓚ (for 'known') queries S to check if the current state has already been visited. Notice that the extension of S is dynamic and it can vary during the evaluation of a formula; while the 'concrete' operation we can apply to S is simple membership.

Other operators can naturally be added, for example:

$$\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle, w \models \textcircled{e}\varphi \;\text{ iff }\; \langle D, (R_r)_{r\in\mathsf{Rel}}, L, \emptyset\rangle, w \models \varphi.$$

I.e., we can use the erase operator ⓔ to completely wipe out the memory S. We have introduced this family of logics, that we called *memory logics*, and investigated its expressive power in [2,3,4].

The language we have just described is very flexible, and it can be used to easily characterize model properties. For example if all states in the domain of a model $\mathcal{M}$ satisfy the formula ⓔⓡ$\langle r\rangle$ⓚ then the relation R_r is reflexive (we wipe out the memory, memorize the current point of evaluation and verify that it is accessible). Similarly, if they satisfy ⓔⓡ$[r]\langle r\rangle$ⓚ then R_r is symmetric (erase memory, memorise the current point of evaluation and verify that all successors can reach back to the memorized point). Actually, using ⓔ, ⓡ and ⓚ we can express properties similarly as how it is done in the hybrid language $\mathcal{HL}(\downarrow)$ [5]. But in [3,4] we have shown that the modal language extended with ⓔ, ⓡ and ⓚ is strictly less expressive than $\mathcal{HL}(\downarrow)$.

In this article we are interested in providing complete axiomatizations for these logics. With this aim in mind, we will include in the language also nominals and the hybrid @ operator (see [5,6] for details on hybrid logics). As discussed in [8], the hybrid machinery can be used to prove general completeness results, and to axiomatize logics which are otherwise difficult to characterize.

The rest of the paper is organized as follows. In the next section we formally introduce the different logics we will investigate. In Sect. 3 we present a sound

and complete axiomatization for $\mathcal{HL}(@, ⓡ, ⓚ)$[1], the basic modal logic extended with nominals, @, and the ⓡ and ⓚ memory operators. Our axiomatization crucially involves the use of nominals. As discussed in [3], the ⓡ and ⓚ operators are very expressive, and even when added to the basic modal language, they already give rise to an undecidable logic. In [3] we introduced a logic including ⓡ and ⓚ with additional constrains on how the modal and the memory operators interact. In that work we showed that although this logic is strictly more expressive than $\mathcal{ML}$, it turns out to be decidable. We will show a sound and complete axiomatization for this logic in Sect. 4. Finally in Sect. 5 we discuss completeness for a language including the ⓔ operator. We conclude in Sect. 6 with some final remarks.

2 Syntax and Semantics of Memory Logics

In this section we formally introduce the languages mentioned above, together with some basic notation and notions related to completeness.

Definition 1 (Syntax). *Let* $\mathsf{Prop} = \{p_1, p_2, \ldots\}$ *(the* propositional symbols*),* $\mathsf{Nom} = \{n_1, n_2, \ldots\}$ *(the* nominal symbols*) and* $\mathsf{Rel} = \{r_1, r_2, \ldots\}$ *(the* relational symbols*) be pairwise disjoint, countable infinite sets. Let* $\mathsf{Atom} = \mathsf{Prop} \cup \mathsf{Nom}$*. The set* Forms *of formulas in the signature* $\langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel} \rangle$ *is defined as:*

$$\mathsf{Forms} ::= \top \mid p \mid i \mid ⓚ \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle r \rangle \varphi \mid @_i \varphi \mid ⓡ\varphi \mid ⓔ\varphi,$$

where $p \in \mathsf{Prop}$*,* $i \in \mathsf{Nom}$*,* $r \in \mathsf{Rel}$ *and* $\varphi, \varphi_1, \varphi_2 \in \mathsf{Forms}$*. We use* $[r]\varphi$ *as a shorthand for* $\neg\langle r \rangle \neg\varphi$*.*

Definition 2 (Semantics). *Given a signature* $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel} \rangle$*, a model for* $\mathcal{S}$ *is a tuple* $\langle D, (R_r)_{r \in \mathsf{Rel}}, L, S \rangle$*, satisfying the following conditions: (i)* $D \neq \emptyset$*; (ii) each* R_r *is a binary relation on* D*; (iii)* $L : \mathsf{Atom} \to 2^D$ *is a labeling function such that* $L(n)$ *is a singleton whenever* $n \in \mathsf{Nom}$*; and (iv)* $S \subseteq D$*.*

Given the model $\mathcal{M} = \langle D, (R_r)_{r \in \mathsf{Rel}}, L, S \rangle$ *and* $w \in D$*, the semantics for the different operators is defined as follows:*

$$\begin{array}{rll}
\mathcal{M}, w \models \top & \textit{always} & \\
\mathcal{M}, w \models p & \textit{iff} & w \in L(p) \quad p \in \mathsf{Atom} \\
\mathcal{M}, w \models \neg\varphi & \textit{iff} & \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \textit{iff} & \mathcal{M}, w \models \varphi \textit{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \langle r \rangle \varphi & \textit{iff} & \textit{there is } w' \textit{ such that } R_r(w, w') \textit{ and } \mathcal{M}, w' \models \varphi \\
\mathcal{M}, w \models @_i \varphi & \textit{iff} & \mathcal{M}, v \models \varphi \textit{ where } L(i) = \{v\} \\
\mathcal{M}, w \models ⓡ\varphi & \textit{iff} & \langle D, (R_r)_{r \in \mathsf{Rel}}, L, S \cup \{w\} \rangle, w \models \varphi \\
\mathcal{M}, w \models ⓚ & \textit{iff} & w \in S \\
\mathcal{M}, w \models ⓔ\varphi & \textit{iff} & \langle D, (R_r)_{r \in \mathsf{Rel}}, L, \emptyset \rangle, w \models \varphi.
\end{array}$$

[1] Our notational convention for naming logics is the following. We will call $\mathcal{ML}$ the basic modal logic, with the standard modal operators. We will use $\mathcal{HL}$ for the modal language extended with only nominals. And we will then list the additional operators included in the language. For example $\mathcal{HL}(@, ⓡ, ⓚ, ⓔ)$ is the modal language extended with nominals, and the @, ⓡ, ⓚ and ⓔ operators.

Given a model $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ *and states* $w_1, \ldots, w_n \in D$*, we will write* $\mathcal{M}[w_1, \ldots, w_n]$ *for the model* $\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S \cup \{w_1, \ldots, w_n\}\rangle$.

Given a model $\mathcal{M}$*, we say that* φ *is* valid on $\mathcal{M}$ *and write* $\mathcal{M} \models \varphi$ *if for all states* w *in the domain of* $\mathcal{M}$ *we have that* $\mathcal{M}, w \models \varphi$.

Definition 3 (Satisfiability, Validity, Completeness). *Let* $\mathcal{C}$ *be a class of models. We say that* φ *is* satisfiable in $\mathcal{C}$ *if there is a model* $\mathcal{M} \in \mathcal{C}$ *and a state* w *in the domain of* $\mathcal{M}$ *such that* $\mathcal{M}, w \models \varphi$*. We say that* φ *is* valid in $\mathcal{C}$ *if* $\neg\varphi$ *is not satisfiable in* $\mathcal{C}$*. The notions of satisfiability and validity can be extended to set of formulas in the usual way. For example, we say that a set of formulas* Γ *is satisfiable in a class of models* $\mathcal{C}$ *if there is a model* $\mathcal{M} \in \mathcal{C}$ *and a state* w *in the domain of* $\mathcal{M}$ *such that for all formulas* $\varphi \in \Gamma$ *we have* $\mathcal{M}, w \models \varphi$*. We will note* $T(\mathcal{C})$ *the set of all valid formulas in* $\mathcal{C}$.

Given an axiomatization $\mathcal{A}$*, a formula* φ *is a* theorem of $\mathcal{A}$ *if it is an axiom in* $\mathcal{A}$*, or it can be obtained by a finite number of applications of inference rules in* $\mathcal{A}$ *from axioms of* $\mathcal{A}$*. We write* $T(\mathcal{A})$ *for the set of all theorems in* $\mathcal{A}$.

We say that a formula φ *is* consistent *with respect to an axiomatization* $\mathcal{A}$ *(or* $\mathcal{A}$*-consistent) if* $\neg\varphi$ *is not a theorem of* $\mathcal{A}$*. The notion of consistency can be extended to a set of formulas* Γ *by requiring that for no finite subset* Γ^f*, the formula* $\bigwedge \Gamma^f \to \neg\top$ *be a theorem of* $\mathcal{A}$.

Given an axiomatization $\mathcal{A}$ *and a class of models* $\mathcal{C}$ *we say that* $\mathcal{A}$ *is* sound for $\mathcal{C}$ *if* $T(\mathcal{A}) \subseteq T(\mathcal{C})$*, and that it is* complete for $\mathcal{C}$ *if* $T(\mathcal{C}) \subseteq T(\mathcal{A})$*. Completeness can be equivalently defined in terms of consistency and satisfiability:* $\mathcal{A}$ *is complete for* $\mathcal{C}$ *if every formula consistent in* $\mathcal{A}$ *is satisfiable in* $\mathcal{C}$.

Finally, we say that an axiomatization $\mathcal{A}$ *is* strongly complete *with respect to* $\mathcal{C}$*, if every* $\mathcal{A}$*-consistent set of formulas is satisfiable in* $\mathcal{C}$.

In this article we will present a number of axiomatizations and prove them (strongly) complete with respect to different classes of models. The different logical languages involved will be defined in terms of the operators introduced in Definitions 1 and 2; and we will be interested mainly in the class of all models, and the class $\{\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle \mid S = \emptyset\}$ of models with no previously 'remembered' states. This last class is a natural choice: in the absence of the $\textcircled{e}$ operator, evaluating formulas on such models provides additional expressivity, and the intuitive meaning of the *remember* and *known* operators are naturally captured. For example the formula $\textcircled{r}\langle r\rangle\textcircled{k}$ characterizes reflexivity of R_r over this class (that is, let $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ be an arbitrary model, except that $S = \emptyset$, then $\mathcal{M} \models \textcircled{r}\langle r\rangle\textcircled{k}$ if and only if R_r is reflexive). This no longer holds when S is arbitrary. See [3] for further details.

As we mentioned in the introduction, we will also be interested in a logic in which the behavior of the *remember* operator is highly coupled with the modal transitions to ensure decidability. In this logic, every time we make a modal step, we are constrained to remember the current state. We change the semantic definition of $\langle r\rangle$ to be:

$$\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle, w \models \langle r\rangle\varphi \text{ iff } \exists w' \in D, R_r(w, w') \text{ and } \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S \cup \{w\}\rangle, w' \models \varphi$$

Axioms:

CT	All classical tautologies	*Intro*	$\vdash (i \wedge p) \to @_i p$
$K_@$	$\vdash @_i(p \to q) \to @_i p \to @_i q$	*Self-dual*$_@$	$\vdash @_i p \leftrightarrow \neg @_i \neg p$
$K_{[r]}$	$\vdash [r](p \to q) \to ([r]p \to [r]q)$	*Ref*	$\vdash @_i i$
Sym	$\vdash @_i j \leftrightarrow @_j i$	*Nom*	$\vdash (@_i j \wedge @_j p) \to @_i p$
Agree	$\vdash @_j @_i p \leftrightarrow @_i p$	*Back*	$\vdash \langle r \rangle @_i p \to @_i p$
Rem	$\vdash @_i(\textcircled{r}\varphi \leftrightarrow \varphi[\textcircled{k}/(\textcircled{k} \vee i)])$		

Rules:

MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$	$Gen_{[r]}$	If $\vdash \varphi$ then $\vdash [r]\varphi$
Name	$\vdash j \to \varphi$ then $\vdash \varphi$ (j not in φ)	$Gen_@$	If $\vdash \varphi$ then $\vdash @_i \varphi$
Paste	If $\vdash (@_i \langle r \rangle j \wedge @_j \varphi) \to \psi$ then $\vdash (@_i \langle r \rangle \varphi) \to \psi$ ($j \neq i$ and j is not in φ or ψ)		
SortedSub$_1$	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in \mathsf{Prop}$		
SortedSub$_2$	If $\vdash \varphi$ then $\vdash \varphi[i/j]$ for any $i, j \in \mathsf{Nom}$		

The expression $\varphi[a/b]$ is the result of uniformly replacing all occurrences of a in φ by b.

Fig. 1. Axiomatization for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$

We call this logic $\mathcal{ML}^-$ ($\mathcal{HL}^-$ for the hybrid case). As we proved in [4], $\mathcal{ML}^-(\textcircled{r}, \textcircled{k})$ is decidable and strictly more expressive than $\mathcal{ML}$.

3 Completeness for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$

This section is devoted to prove a completeness result for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$. Our axiomatization is shown in Fig. 1. It is an extension of the axiomatization for $\mathcal{HL}(@)$ presented in [7].

The axiom characterizing the behavior of the memory operator is *Rem*. To show soundness of the axiomatization, we only have to look at this new axiom. Intuitively, the axiom says that, when standing in a state named by i, the act of remembering the current state is equivalent to increase the extension of $\textcircled{k}$ with i throughout the formula. Formally:

Lemma 1. *Let $\mathcal{M}$ be a model and $w \in \mathcal{M}$ such that $\mathcal{M}, w \models i$. Then, for all $v \in \mathcal{M}$, $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi[\textcircled{k}/(\textcircled{k} \vee i)]$.*

Proof. By induction on φ. For the base case, if φ is a proposition symbol or a nominal, then since $\varphi = \varphi[\textcircled{k}/(\textcircled{k} \vee i)]$ we have $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi$. For the $\textcircled{k}$ case we have to prove $\mathcal{M}[w], v \models \textcircled{k}$ iff $\mathcal{M}, v \models \textcircled{k} \vee i$.

$\Rightarrow$) Assume that $\mathcal{M}[w], v \models \textcircled{k}$. If $v = w$, then $\mathcal{M}, v \models i$, and therefore $\mathcal{M}, v \models \textcircled{k} \vee i$. If $v \neq w$, then $\mathcal{M}, v \models \textcircled{k}$, and hence $\mathcal{M}, v \models \textcircled{k} \vee i$.

$\Leftarrow$) Let's assume that $\mathcal{M}, v \models \textcircled{k} \vee i$. If $v = w$, then $\mathcal{M}[w], v \models \textcircled{k}$. On the other hand, if $v \neq w$, then we know that $\mathcal{M}[w], v \models \neg i$, and therefore $\mathcal{M}, v \models \textcircled{k}$. We conclude $\mathcal{M}[w], v \models \textcircled{k}$.

The conjunction, negation, diamond, @ and remember cases are straightforward, using the inductive hypothesis and the fact that the replacement operation $[\textcircled{k}/(\textcircled{k} \vee i)]$ distributes over $\wedge$, $\neg$, $\langle r \rangle$, @ and $\textcircled{r}$.

Corollary 1. Rem *is sound over the class of all models.*

Proof. Take an arbitrary model $\mathcal{M}$ and let $w \in \mathcal{M}$ be such that $\mathcal{M}, w \models i$. By definition $\mathcal{M}, v \models @_i \textcircled{r}\varphi$ iff $\mathcal{M}[w], w \models \varphi$. Applying the previous lemma, this happens iff $\mathcal{M}, w \models \varphi[\textcircled{k}/(\textcircled{k} \vee i)]$ iff (by definition) $\mathcal{M}, v \models @_i\varphi[\textcircled{k}/(\textcircled{k} \vee i)]$.

It is worth noting that having nominals in the language is a key feature to describe the $\textcircled{r}/\textcircled{k}$ interaction with modal operators, and the *Rem* axiom strongly uses this feature. The possibility to identify with a nominal the state in which a remember operation is taking place allows us to fully describe the behavior of this interaction.

We now turn to completeness. We will build a Henkin model using named maximal consistent sets (MCSs) for an arbitrary consistent set (see [7] for further details).

Definition 4. *An* MCS *is* named *if and only if it contains a nominal. We call any nominal belonging to an* MCS *a* name *for that* MCS*. Also, if Γ is an* MCS *and i is a nominal, then we call $\{\varphi \mid @_i\varphi \in \Gamma\}$* a named set yielded by *Γ. Furthermore we say that a model is named if every state in the model is the denotation of some nominal (for all $w \in D$ there is some nominal i such that $L(i) = \{w\}$).*

The idea behind the construction presented in [7] is that we can extract all the information we need to build a named canonical model from a single MCS. We start by noting that hidden inside any MCS there is a collection of named MCSs with a number of relevant properties:

Lemma 2. *Let Γ be an* MCS*. For every nominal i, let Δ_i be $\{\varphi \mid @_i\varphi \in \Gamma\}$. Then, (i) for every nominal i, Δ_i is an* MCS *that contains i; (ii) for all nominals i and j, if $i \in \Delta_j$, then $\Delta_i = \Delta_j$; (iii) for all nominals i and j, $@_i\varphi \in \Delta_j$ iff $@_i\varphi \in \Gamma$; and (iv) if i is a name for Γ then $\Gamma = \Delta_i$.*

Proof. We only sketch the proof, the full details can be found in [7]. Claim (i) can be proved using *Ref* (to guarantee that $i \in \Delta_i$), $Gen_@$ and *Self-dual*$_@$ (to prove that Δ_i is an MCS). Claim (ii) is proved using *Sym* and *Nom*, Claim (iii) follows by *Agree*. And Claim (iv) is obtained by *Intro* and *Self-dual*$_@$.

Given a consistent set of formulas Σ, we can always expand it to an MCS Σ^+ using the standard Lindenbaum's Lemma. The problem is that nothing guarantees that this MCS will be named. In addition, as we want to extract named MCSs from named sets yielded by Σ^+, we have to ensure that there are enough named MCSs to use as existential witnesses during the construction of the Henkin model. Here is where the *Name* and *Paste* rules are useful. Expanding the language with new nominals, the *Name* rule is going to solve our first problem, and the *Paste* rule solves the second. We call an MCS Γ *pasted* iff $@_i\langle r\rangle\varphi \in \Gamma$ implies that for some nominal j, $@_i\langle r\rangle j \wedge @_j\varphi \in \Gamma$. *Name* and *Paste* guarantee that any consistent set of formulas can be extended to a named and pasted MCS.

Lemma 3 (Extended Lindenbaum Lemma). *Let $\mathcal{S} = \langle \mathsf{Prop}, \mathsf{Nom}, \mathsf{Rel}\rangle$ be a signature, let Nom' be a countably infinite collection of nominals disjoint from Nom, and let $\mathcal{S}'$ be the signature obtained by extending $\mathcal{S}$ with Nom'. Then every $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$-consistent set of formulas in S can be extended to a named and pasted MCS in $\mathcal{S}'$.*

Proof. Full details can be found in [7]. The proof follows the standard Lindenbaum's construction with the following modifications. Take a consistent set of formulas Σ, and name it by adding a new nominal k (use *Name* to prove consistency). Using an enumeration of all the formulas, we expand Σ step-by-step with a formula that is consistent with the expanded set at each point. Because we want the final MCS to be pasted, at the $(m+1)$-th step, when we are considering Σ^m and the formula φ_{m+1}, if $\Sigma^m \cup \{\varphi_{m+1}\}$ is inconsistent, we set $\Sigma^{m+1} = \Sigma^m$. Else, if φ_{m+1} has the form $@_i\langle r\rangle\varphi$, we set $\Sigma^{m+1} = \Sigma^m \cup \{\varphi_{m+1}\} \cup \{@_i\langle r\rangle j \wedge @_j\langle r\rangle\varphi\}$, where j is new (relying on the *Paste* rule for consistency). If φ_{m+1} does not have the form $@_i\langle r\rangle\varphi$, we set $\Sigma^{m+1} = \Sigma^m \cup \{\varphi_{m+1}\}$ as usual. Finally, we take the infinite union of all the Σ^i.

Now we can define the model we need, using the named sets yielded by a named and pasted MCS.

Definition 5. *Let Γ be a named and pasted MCS. The* named model yielded by *Γ is $\mathcal{M}^\Gamma = (D^\Gamma, (R_r^\Gamma)_{r\in\mathsf{Rel}}, L^\Gamma, S^\Gamma)$. Here D^Γ is the set of all named sets yielded by Γ, $R_r^\Gamma(u,v)$ holds iff for all formulas φ, $\varphi \in v$ implies $\langle r\rangle\varphi \in u$, $L^\Gamma(a) = \{w \in W^\Gamma \mid a \in w\}$ for any atom a, and $S^\Gamma = \{w \mid \textcircled{k} \in w\}$.*

Note that $\mathcal{M}^\Gamma$ is a well defined model, since by items (i) and (ii) of Lemma 2, L^Γ assigns to every nominal a singleton subset of D^Γ. Using the fact that Γ is named and pasted, we can prove the following Existence Lemma

Lemma 4 (Existence Lemma [7]). *Let Γ be a named and pasted MCS, and let $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ be the named model yielded by Γ. Suppose $u \in \mathcal{M}$ and $\langle r\rangle\varphi \in u$. Then there is a $v \in \mathcal{M}$ such that $R_r(u,v)$ and $\varphi \in v$*

Now we are ready to prove the Truth Lemma that will lead us to the desired completeness result. Before that, to treat the $\textcircled{r}$ case properly, we have to redefine the complexity of the formulas, to be able to handle the substitutions made by the *Rem* axiom.

Definition 6. *We define the complexity of a formula as $comp(\varphi) = 2(k+1)(r+1)(d+1)+v$, where k, r and d are the number of occurrences of $\textcircled{k}$, $\textcircled{r}$ and $\langle r\rangle$ respectively, and v is the number of occurrences of all the other possible symbols.*

Note that with this definition, $comp(\textcircled{r}\varphi) > comp(\varphi[\textcircled{k}/(\textcircled{k} \vee i)])$.

Lemma 5 (Truth Lemma). *Let $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ be the named model yielded by a named and pasted MCS, and let $u \in D$. Then, for all formulas φ, $\varphi \in u$ iff $\mathcal{M}, u \models \varphi$.*

Proof. By Induction on the structure of φ. The atomic, boolean and modal cases are obvious (the Existence Lemma is used for the modal case, and the $\textcircled{k}$ case follows directly from the definition of S^{Γ}). We analyze the satisfaction operators. Suppose $\mathcal{M}, u \models @_i\psi$. This happens iff $\mathcal{M}, \Delta_i \models \psi$ (by items (i) and (ii) of Lemma 2, Δ_i is the only MCS containing i, and hence, by the atomic case of the present lemma, the only state in $\mathcal{M}$ where i is true) iff $\psi \in \Delta_i$ (by inductive hypothesis) iff $@_i\psi \in \Delta_i$ (using the fact that $i \in \Delta_i$ together with *Intro* for the left-to-right direction and *Intro* and *Self-dual*$_@$ for the right-to-left direction) iff $@_i\psi \in u$ (by *Agree*).

To finish let's analyze the case for $\textcircled{r}$. Given $u \in \mathcal{M}$, we know that for some nominal i we have $u = \Delta_i$, so by definition, $\mathcal{M}, u \models i$ and $i \in u$. Suppose $\mathcal{M}, u \models \textcircled{r}\psi$. This happens iff $\mathcal{M}, u \models @_i\textcircled{r}\psi$ (because $\mathcal{M}, u \models i$) iff $\mathcal{M}, u \models @_i\psi[\textcircled{k}/(\textcircled{k}\vee i)]$ (by Corollary 1) iff $\mathcal{M}, u \models \psi[\textcircled{k}/(\textcircled{k}\vee i)]$ (again because $\mathcal{M}, u \models i$) iff $\psi[\textcircled{k}/(\textcircled{k}\vee i)] \in u$ (by inductive hypothesis) iff $@_i\psi[\textcircled{k}/(\textcircled{k}\vee i)] \in u$ (because $i \in u$, using *Intro* for the left-to-right direction, and *Self-dual*$_@$ and *Intro* for the right-to-left direction) iff $@_i\textcircled{r}\psi \in u$ (by the *Rem* axiom) iff $\textcircled{r}\psi \in u$ (because $i \in u$, applying again *Intro* and *Self-dual*$_@$).

Theorem 1 (Completeness for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$). *Every* MCS *of formulas in* $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$ *is satisfiable in a countable named model.*

Proof. Let Σ be a consistent set of formulas from $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$. We use the Extended Lindenbaum Lemma to expand it to a named and pasted set Σ^+ in an extended countable language. Let $\mathcal{M}$ be the named model yielded by Σ^+. By item (iv) of Lemma 2, because Σ^+ is named, Σ^+ is an element in the domain of $\mathcal{M}$. By the Truth Lemma, $\mathcal{M}, \Sigma^+ \models \Sigma$. The model is countable because each state is named by some nominal in the extended language, and there are only countably many of these.

This establishes strong completeness as desired. But in fact, we have done more. Because our Henkin model is *named*, we can prove a more general result.

Definition 7. *If a formula φ contains no propositional symbols (that is, its atoms are nominals or $\textcircled{k}$), we say that φ is* $\textcircled{k}$-pure. *Furthermore, if φ is a $\textcircled{k}$-pure formula, we say that ψ is a* $\textcircled{k}$-pure instance *of φ if ψ is obtained from φ by uniformly substituting nominals for nominals. A formula φ is* pure *if its atomic subformulas are only nominals.*

The axiomatization we presented in Fig. 1 for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$ has the following property: for any set of pure formulas Π, if P is the logic obtained by adding the formulas in Π as axioms, then P is complete with respect to the class defined by Π.[2] This result can be extended to $\textcircled{k}$-pure formulas for the case of $\mathcal{HL}_{\emptyset}(@, \textcircled{r}, \textcircled{k})$, the logic obtained over the class $\{\langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle \mid S = \emptyset\}$ of models with no previously remembered states.

We first state a property that will help us achieve the completeness result for pure axioms.

[2] These general completeness results are standard in hybrid logics. See [8] for further details.

Lemma 6. *Let $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ be a named model.*

1. *Let φ be a pure formula, and suppose that for all pure instances ψ of φ, $\mathcal{M} \models \psi$. Then for any L' and S', $\langle D, (R_r)_{r\in\mathsf{Rel}}, L', S'\rangle \models \varphi$.*
2. *Let $S = \emptyset$, φ be a $\textcircled{k}$-pure formula, and suppose that for all $\textcircled{k}$-pure instances ψ of φ, $\mathcal{M} \models \psi$. Then for any L', $\langle D, (R_r)_{r\in\mathsf{Rel}}, L', S\rangle \models \varphi$.*

Proof. We only discuss item 2. Suppose that the hypothesis hold, but for some labeling L', $\langle D, (R_r)_{r\in\mathsf{Rel}}, L', \emptyset\rangle \not\models \varphi$. We can take ρ, a $\textcircled{k}$-pure instance of φ, such that ρ is obtained from φ replacing each nominal i by j, where $L'(i) = L(j)$. By an induction on the formula complexity, it is easy to see that $(D, (R_r)_{r\in\mathsf{Rel}}, L, \emptyset) \not\models \rho$. This is a contradiction.

With the help of Lemma 6, and since we showed that we can build named models from $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$-MCSs, a wide range of strong completeness results can be established.

Theorem 2 ([7]). *Let Π be a set of pure formulas and let $\mathcal{A}$ be the axiomatization obtained by adding formulas in Π as axioms to the axiomatization shown in Fig. 1. Then, every $\mathcal{A}$-consistent set of formulas is satisfiable in a countable named model in the class defined by Π.*

Proof. Given an $\mathcal{A}$-consistent set of formulas Ω, we can use the Extended Lindenbaum's Lemma to extend it to a named an pasted $\mathcal{A}$-MCS Ω^+. The named model $\mathcal{M}^\Omega$ that Ω^+ gives rise to will satisfy Ω at Ω^+. In addition, as every formula in Π belongs to every $\mathcal{A}$-MCS, we have that $\mathcal{M}^\Omega \models \Pi$. Therefore, by Lemma 6, $\mathcal{M}^\Omega$ is in the class of models defined by Π.

To finish this section, we will discuss an extension of the axiomatization presented above, to characterize $\mathcal{HL}_\emptyset(@, \textcircled{r}, \textcircled{k})$.

Theorem 3. *The system obtained by extending the axiomatization in Fig. 1 with the axiom $(Empty) \vdash \neg\textcircled{k}$ is sound and strongly complete for $\mathcal{HL}_\emptyset(@, \textcircled{r}, \textcircled{k})$.*

Proof. Soundness of *Empty* is obvious for the class of $\mathcal{HL}_\emptyset(@, \textcircled{r}, \textcircled{k})$-models. The completeness proof is as the one for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$, but in addition, thanks to *Empty*, all maximal consistent sets Δ_i are such that $\neg\textcircled{k} \in \Delta_i$. Therefore, the final model yielded by Γ, $\mathcal{M}^\Gamma = \langle D^\Gamma, (R_r^\Gamma)_{r\in\mathsf{Rel}}, L^\Gamma, S^\Gamma\rangle$, is such that $S^\Gamma = \emptyset$, and thus, it is a $\mathcal{HL}_\emptyset(@, \textcircled{r}, \textcircled{k})$-model.

Proposition 1. *For the case of $\mathcal{HL}_\emptyset(@, \textcircled{r}, \textcircled{k})$, the result of adding Π, a set of pure formulas, can be extended to a set Π of $\textcircled{k}$-pure formulas*

Proof. Trivial, using Lemma 6, and the same proof as in Theorem 3.

4 The Case for $\mathcal{ML}^-(\textcircled{r}, \textcircled{k})$

We will present a sound and complete axiomatization for $\mathcal{ML}^-(\textcircled{r}, \textcircled{k})$. In the previous section we mentioned the importance of nominals to describe the interaction between memory and modal operators. In this section we will show that if we

restrict ourselves to the logic in which we are constrained to remember the current state every time we make a modal transition, nominals can be avoided. In this logic we can describe the interaction between Ⓡ and Ⓚ at a propositional level. This is not a coincidence. Because this logic has the tree model property [3,4], we can asume that we evaluate $\mathcal{ML}^-(Ⓡ, Ⓚ)$-formulas on trees, and since there are no cycles, the remember operator has no real effect beyond the current state.

Given a formula φ, we define the formula $\varphi^\sharp$ as the result of replacing all the occurrences of Ⓚ that are in φ at modal depth zero by $\top$. Formally:

$$\begin{aligned} p^\sharp &= p \quad p \in \mathsf{Prop} \\ Ⓚ^\sharp &= \top \\ (\neg\varphi)^\sharp &= \neg\varphi^\sharp \\ (\varphi_1 \wedge \varphi_2)^\sharp &= \varphi_1^\sharp \wedge \varphi_2^\sharp \\ (Ⓡ\varphi)^\sharp &= Ⓡ\varphi^\sharp \\ (\langle r\rangle\varphi)^\sharp &= \langle r\rangle\varphi \end{aligned}$$

Lemma 7. *$\mathcal{M}, w \models Ⓡ\varphi$ iff $\mathcal{M}, w \models \varphi^\sharp$.*

Proof. We proceed by induction. The case for Ⓚ, the propositional symbols and boolean connectives are straightforward. We analyze the other cases. For the case $\varphi = Ⓡ\psi$. $\mathcal{M}, w \models ⓇⓇ\psi$ iff $\mathcal{M}, w \models Ⓡ\psi$ iff (by inductive hypothesis) $\mathcal{M}, w \models \psi^\sharp$ iff $\mathcal{M}, w \models (\psi^\sharp)^\sharp$ iff (by inductive hypothesis) $\mathcal{M}, w \models Ⓡ(\psi^\sharp)$ iff $\mathcal{M}, w \models (Ⓡ\psi)^\sharp$. For the case $\varphi = \langle r\rangle\psi$. $\mathcal{M}, w \models Ⓡ\langle r\rangle\psi$ iff (by definition) $\mathcal{M}[w], w \models \langle r\rangle\psi$ iff (by definition of $\langle r\rangle$) there is a $v \in \mathcal{M}$, $R_r(w, v)$ such that $\mathcal{M}[w], v \models \psi$ iff (by definition of $\langle r\rangle$) $\mathcal{M}, w \models \langle r\rangle\psi$ iff (by definition of $\sharp$) $\mathcal{M}, w \models (\langle r\rangle\psi)^\sharp$.

We are now ready to present the axiomatization. The axiomatization for $\mathcal{ML}^-(Ⓡ, Ⓚ)$ (shown in Fig. 2) is an extension of the axiomatization for the basic modal logic [7], plus the axiom $Rem^- \vdash Ⓡ\varphi \leftrightarrow \varphi^\sharp$.

Soundness of Rem^- follows from Lemma 7. We will prove completeness with respect to the class of acyclic models, and therefore for the class of all models. We will use a step-by-step construction. I.e., instead of building the entire canonical model, we will carry out a stepwise selection from MCSs of the canonical model of $\mathcal{ML}^-(Ⓡ, Ⓚ)$ as our basic building blocks.[3]

We define $\mathcal{M}^c = \langle D^c, (R_r^c)_{r\in\mathsf{Rel}}, L^c, S^c\rangle$, the $\mathcal{ML}^-(Ⓡ, Ⓚ)$ canonical model, in the usual sense (see [7] for details). That is, D^c is the set of all maximal consistent sets of formulas of $\mathcal{ML}^-(Ⓡ, Ⓚ)$, $R_r^c(\Gamma, \Delta)$ iff for all $\varphi \in \Delta$, $\langle r\rangle\varphi \in \Gamma$, $\Gamma \in L^c(p)$ iff $p \in \Gamma$ and $S^c = \{\Gamma \mid Ⓚ \in \Gamma\}$.

Definition 8. *A network $\mathcal{N} = \langle N, (R_r)_{r\in\mathsf{Rel}}, v\rangle$ is a triple where N is a countable non-empty set of elements, each R_r is a binary relation on N, and v is a function that maps elements in N to maximal consistent sets.*

We say that a network is coherent *if (C1) $\bigcup_{r\in\mathsf{Rel}} R_r$ defines an acyclic graph and (C2) $R_r^c(v(s), v(t))$ for all $s, t \in N$ such that $R_r(s, t)$. A network is* saturated

[3] Alternatively, one can take the standard canonical model and then unravel it to obtain a tree, and therefore acyclic, model.

Axioms:	
CT	All classical tautologies
$K_{[r]}$	$\vdash [r](p \to q) \to ([r]p \to [r]q)$
Rem^-	$\vdash \textcircled{r}\varphi \leftrightarrow \varphi^\sharp$
Rules:	
MP	If $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$
$Gen_{[r]}$	If $\vdash \varphi$ then $\vdash [r]\varphi$
Sub	If $\vdash \varphi$ then $\vdash \varphi[p/\psi]$ for any $p \in \mathsf{Prop}$

Fig. 2. Axiomatization for $\mathcal{ML}^-(\textcircled{r}, \textcircled{k})$

if whenever $\langle r\rangle\psi \in v(s)$ *for some* $s \in N$*, then there is a* $t \in N$ *such that* $R_r(s,t)$ *and* $\psi \in v(t)$*.*

We want networks to play the role of models, so we have to check that we have imposed the right conditions on a network to achieve this.

Definition 9. *Let* $\mathcal{N} = \langle N, (R_r)_{r\in\mathsf{Rel}}, v\rangle$ *be a network. We define the induced labeling* $L_\mathcal{N}(p) = \{s \in N \mid p \in v(s)\}$*, the induced set of remembered states* $S_\mathcal{N} = \{s \in N \mid \textcircled{k} \in v(s)\}$*, and the induced model* $\mathcal{M}_\mathcal{N} = \langle N, (R_r)_{r\in\mathsf{Rel}}, L_\mathcal{N}, S_\mathcal{N}\rangle$*.* $\mathcal{F}_\mathcal{N} = \langle N, (R_r)_{r\in\mathsf{Rel}}\rangle$ *is called the* underlying frame *of* $\mathcal{N}$*.*

We are now ready to prove a Truth Lemma.

Lemma 8 (Truth Lemma). *Let* $\mathcal{N} = \langle N, (R_r)_{r\in\mathsf{Rel}}, v\rangle$ *be a coherent and saturated network. Then, for all* φ *and* $s \in N$*,*

$$\mathcal{M}_\mathcal{N}, s \models \varphi \;\; \textit{iff} \;\; \varphi \in v(s).$$

Proof. Before we prove this lemma, let us observe the following property: let $\mathcal{M} = \langle D, (R_r)_{r\in\mathsf{Rel}}, L, S\rangle$ be an acyclic model, and let $w, v \in D$ be such that $R_r(w,v)$. Then for all formulas φ, $\mathcal{M}[w], v \models \varphi$ iff $\mathcal{M}, v \models \varphi$.

We now proceed by induction on φ. The propositional case, the $\textcircled{k}$ case and the boolean cases are straightforward, given the definition of $\mathcal{M}_\mathcal{N}$. Let's suppose that $\mathcal{M}_\mathcal{N}, s \models \textcircled{r}\psi$. This happens iff (by Lemma 7) $\mathcal{M}_\mathcal{N}, s \models \psi^\sharp$ iff (by inductive hypothesis) $\psi^\sharp \in v(s)$ iff (by Rem^- axiom) $\textcircled{r}\psi \in v(s)$.

The $\langle r\rangle$ case: for the left-to-right direction, if $\mathcal{M}_\mathcal{N}, s \models \langle r\rangle\psi$, then there exists $t \in N$ such that $R_r(s,t)$ and $\mathcal{M}_\mathcal{N}[s], t \models \psi$. Therefore, $\mathcal{M}_\mathcal{N}, t \models \psi$. By inductive hypothesis, $\psi \in v(t)$. Because the network is coherent, and $R_r(s,t)$, then $R_r^c(v(s), v(t))$, and we conclude $\langle r\rangle\psi \in v(s)$. For the other direction, let's suppose that $\langle r\rangle\psi \in v(s)$. Because the network is saturated, there is a $t \in N$ such that $\psi \in v(t)$ and $R_r(s,t)$. By inductive hypothesis, $\mathcal{M}_\mathcal{N}, t \models \psi$, so $\mathcal{M}_\mathcal{N}[s], t \models \psi$, and therefore by definition, $\mathcal{M}_\mathcal{N}, s \models \langle r\rangle\psi$.

Summing up, we have reduced the problem of finding a model for an $\mathsf{MCS}\ \Delta$ to a search for a coherent and saturated network for Δ. The idea here is to start with a coherent network and, one step at a time, remove the defects that are preventing the network from being saturated.

Definition 10. *Let $\mathcal{N}$ be a network. We say that $\mathcal{N}$ has a saturation defect if there is a node $s \in N$ and a formula $\langle r \rangle \psi \in v(s)$ such that there is not a $t \in N$, $R(s,t)$ and $\psi \in v(t)$.*

Because a coherent network may have saturation defects, we have to say more about what is the meaning of repairing a defect. We are going to *extend* a network with a saturation defect with another where the defect is corrected.

Definition 11. *Let $\mathcal{N}_0 = \langle N_0, R_0, v_0 \rangle$ and $\mathcal{N}_1 = \langle N_1, R_1, v_1 \rangle$ be two networks. We say that $\mathcal{N}_1$ extends $\mathcal{N}_0$ if $\mathcal{F}_{\mathcal{N}_0}$ is a subframe of $\mathcal{F}_{\mathcal{N}_1}$ and v_0 agrees with v_1 on N_0.*

The following lemma states that a saturation defect of a finite coherent network can always be repaired.

Lemma 9 (Repair Lemma). *Let $\mathcal{N}$ be a finite and coherent network with a saturation defect. Then there is a network $\mathcal{N}'$ extending $\mathcal{N}$ without that defect.*

Proof. Because $\mathcal{N}$ has a a saturation defect, there is a node $s \in N$ and a formula $\langle r \rangle \psi \in v(s)$ such that there is not a $t \in N$, $R_r(s,t)$ and $\psi \in v(t)$. We define $\mathcal{N}'$ as

$$\begin{array}{ll} N' = N \cup \{s'\} & \text{with } s' \notin N \\ R'_r = R_r \cup \{(s,s')\} & \\ v' \ = v \cup \{(s', \Delta)\} & \end{array}$$

where Δ is an MCS containing ψ such that $R^c_r(v(s), \Delta)$ (the existence of such Δ can be proved through an Existence Lemma similar to Lemma 4). Clearly, $\mathcal{N}'$ is a coherent network extending $\mathcal{N}$ and does not have the previous defect.

Now we can prove the desired strong completeness result. We start with a singleton network, and we extend it step by step to a larger network using the Repair Lemma. We obtain the saturated network we want by taking the union of our sequence of networks.

Theorem 4. *The axiomatization is strongly complete with respect to the class of $\mathcal{ML}^-(\textcircled{r}, \textcircled{k})$ models.*

Proof. Let $S = \{s_i \mid i \in \omega\}$. Enumerate the potential saturation defects (the set $S \times \mathsf{Forms}$). Given a consistent set Σ, expand it to an MCS Σ^+. The initial network is $\mathcal{N}^0 = \langle \{s_0\}, \emptyset, (s_0, \Sigma^+) \rangle$, which is finite and coherent. Given a network $\mathcal{N}^i$, $i \geq 0$, where the minimal saturation defect is D, we define $\mathcal{N}^{i+1}$ as the extension of $\mathcal{N}^i$ (following the Repair Lemma) without that defect. If $\mathcal{N}^i$ has no saturation defects, then $\mathcal{N}^{i+1} = \mathcal{N}^i$. Let $\mathcal{N}^\omega = \langle N, (R_r)_{r \in \mathsf{Rel}}, v \rangle$ be:

$$N = \bigcup_{n \in \omega} N^n \qquad R_r = \bigcup_{n \in \omega} R^n_r \qquad v = \bigcup_{n \in \omega} v^n.$$

It is clear that $\mathcal{N}^\omega$ is saturated. For suppose not; let d be the minimal saturation defect (with respect to the enumeration) of $\mathcal{N}^\omega$, say $d = d_k$. By construction, there must be an approximation $\mathcal{N}^i$ of $\mathcal{N}^\omega$ of which d is also a defect. There only can be k defects that are less than d, so d will be repaired before the stage $k + i$ of the construction. This is a contradiction, so $\mathcal{N}^\omega$ is a coherent and saturated network, and therefore $\mathcal{M}_{\mathcal{N}^\omega}, s_0 \models \Sigma$.

Axioms:
All the axioms from $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$ except *Rem*
Rem' $\vdash @_i(\textcircled{r}\varphi \leftrightarrow \varphi_i^\star)$
$Erase_1 \vdash \textcircled{e}\neg\textcircled{k}$
$Erase_2 \vdash \textcircled{e}s \leftrightarrow s \quad s \in \mathsf{Prop} \cup \mathsf{Nom}$
$Erase_3 \vdash \textcircled{e}\neg p \leftrightarrow \neg\textcircled{e}p$
$Erase_4 \vdash \textcircled{e}(p \wedge q) \leftrightarrow (\textcircled{e}p \wedge \textcircled{e}q)$
$Erase_5 \vdash \textcircled{e}\langle r\rangle p \leftrightarrow \langle r\rangle\textcircled{e}p$
$Erase_6 \vdash \textcircled{e}@_i p \leftrightarrow @_i\textcircled{e}p$
$Erase_7 \vdash @_i(\textcircled{e}\textcircled{r}\varphi \leftrightarrow \textcircled{e}\varphi_i^\star)$
Rules:
All the rules from $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$

Fig. 3. Axiomatization for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k}, \textcircled{e})$

5 The Erase Operator

In this section we present an axiomatization for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k}, \textcircled{e})$, taking as a starting point the axiomatization for $\mathcal{HL}(@, \textcircled{r}, \textcircled{k})$ presented in Fig. 1. The first thing we should notice is that the *Rem* axiom is no longer sound. For example, take the valid formula $@_i\textcircled{e}(\textcircled{k} \vee i)$ and use *Rem* to conclude $@_i\textcircled{r}\textcircled{e}\textcircled{k}$. This is clearly a contradiction, since after wiping out the memory, $\textcircled{k}$ cannot be true. Observe that the problem lays in the interaction between $\textcircled{r}$ and $\textcircled{e}$. The replacement operation defined by *Rem* cannot be carried out throughout the whole formula: it should avoid replacements within the scope of an $\textcircled{e}$. More formally, for each formula φ and nominal i we define the formula $\varphi_i^\star$ as follows:

$$
\begin{aligned}
p_i^\star &= p \quad p \in \mathsf{Prop} \cup \mathsf{Nom} \\
\textcircled{k}_i^\star &= \textcircled{k} \vee i \\
(\neg\varphi)_i^\star &= \neg\varphi_i^\star \\
(\varphi_1 \wedge \varphi_2)_i^\star &= {\varphi_1}_i^\star \wedge {\varphi_2}_i^\star \\
(\textcircled{r}\varphi)_i^\star &= \textcircled{r}\varphi_i^\star \\
(\langle r\rangle\varphi)_i^\star &= \langle r\rangle\varphi_i^\star \\
(@_j\varphi)_i^\star &= @_j\varphi_i^\star \\
(\textcircled{e}\varphi)_i^\star &= \textcircled{e}\varphi
\end{aligned}
$$

Analogously to Lemma 1, we can use $(\cdot)^\star$ to characterize the behavior of the $\textcircled{r}$ operator and its interaction with the $\textcircled{e}$ operator.

Lemma 10. *Let $\mathcal{M}$ be a model and $w \in \mathcal{M}$ such that $\mathcal{M}, w \models i$. Then $\mathcal{M}, w \models \textcircled{r}\varphi$ iff $\mathcal{M}, w \models \varphi_i^\star$.*

This result naturally suggests an axiom *Rem'* (shown in Fig. 3) that replaces *Rem*. To characterize the $\textcircled{e}$ operator we should notice first that it behaves

globally and that it does not change the evaluation point. This implies that there is no interaction between Ⓔ and $\neg, \wedge, \langle r \rangle$ and @. To describe the interaction between Ⓔ and Ⓡ we can again make use of the operation $(\cdot)^\star$. The detailed axiomatization is in Fig. 3.

Soundness of this axiomatization is straightforward. The completeness proof uses the same techniques introduced in Sect. 3. The proof of the Truth Lemma is carried out by induction in the structure of the formula, and the new axioms handle the case for Ⓔ by appropriately reducing the complexity.

6 Conclusions and Further Work

In this paper we presented several axiomatizations for some members of the memory logic family. We showed how nominals can be an effective tool to achieve completeness: by allowing to describe the precise interaction between Ⓡ and Ⓚ we could give a completeness result for $\mathcal{HL}(@, Ⓡ, Ⓚ)$. Small variations of this axiomatization leads us to completeness results for other languages, as we showed for $\mathcal{HL}_\emptyset(@, Ⓡ, Ⓚ)$ and $\mathcal{HL}(@, Ⓡ, Ⓚ, Ⓔ)$. Our intention was to give the basic techniques to characterize memory operators using nominals, and not to exhaustively list all possible languages. Observe that, for example, the logic $\mathcal{HL}^-(@, Ⓡ, Ⓚ)$ can be easily axiomatized by replacing the *Back* axiom presented in Fig. 1 by $\vdash @_i\langle r \rangle @_j\varphi \rightarrow @_j\varphi[Ⓚ/(Ⓚ \vee i)]$ (and similarly with the *Paste* rule).

We also showed that nominals are not needed when we add constraints on how $\langle r \rangle$ interacts with Ⓡ, giving a completeness result for $\mathcal{ML}^-(Ⓡ, Ⓚ)$. The idea behind this result lays in the fact that $\mathcal{ML}^-(Ⓡ, Ⓚ)$ has the tree model property and hence, we can describe the interaction between Ⓡ and Ⓚ at a propositional level, independently of the modal operators.

We have not yet found suitable axiomatizations for certain memory logics. Languages without the tree model property, and which do not have nominals seem to be particularly hard to axiomatize. For example, we have not yet been able to devise complete axiomatizations for $\mathcal{ML}(Ⓡ, Ⓚ)$ and $\mathcal{ML}^-_\emptyset(Ⓡ, Ⓚ)$.

We are also interested in other memory operators, besides the ones presented in this paper. A particularly interesting case is the *forget* operator Ⓕ, a local version of the erase operator. While the Ⓔ operator has a global behavior, setting the memory set S to $\emptyset$, we could conceive a local version which only eliminates the current point of evaluation from S.

$$\langle D, (R_r)_{r \in \mathsf{Rel}}, L, S \rangle, w \models Ⓕ\varphi \text{ iff } \langle D, (R_r)_{r \in \mathsf{Rel}}, L, S \setminus \{w\} \rangle, w \models \varphi$$

We have proved that the logic $\mathcal{HL}(@, Ⓡ, Ⓕ, Ⓚ)$ is strictly more expressive than $\mathcal{HL}(@, Ⓡ, Ⓚ)$, but we don't yet have a complete axiomatization. Even having nominals present in the language, we don't know how to characterize the interaction between Ⓡ, Ⓕ and Ⓚ.

References

1. Alur, R., Henzinger, T.: A really temporal logic. Journal of the ACM, 164–169 (1989)
2. Areces, C.: Hybrid logics: The old and the new. In: Proceedings of LogKCA 2007, San Sebastian, Spain (2007)
3. Areces, C., Figueira, D., Figueira, S., Mera, S.: Expressive power and decidability for memory logics. In: Hodges, W., de Queiroz, R. (eds.) Logic, Language, Information and Computation. LNCS, vol. 5110, pp. 56–68. Springer, Heidelberg (2008)
4. Areces, C., Figueira, D., Figueira, S., Mera, S.: Expressive power and decidability for memory logics. Journal of Computer and System Sciences (submitted, 2008); Extended version of [3]
5. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., Wolter, F., van Benthem, J. (eds.) Handbook of Modal Logics. Elsevier, Amsterdam (2006)
6. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Logic Journal of the IGPL 8(3), 339–625 (2000)
7. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2001)
8. Blackburn, P., Tzakova, M.: Hybrid completeness. Logic Journal of the IGPL 6(4), 625–650 (1998)
9. Gerbrandy, J.: Bisimulations on Planet Kripke. PhD thesis, University of Amsterdam, ILLC Dissertation series DS-1999-01 (1999)
10. Harel, E., Lichtenstein, O., Pnueli, A.: Explicit clock temporal logic. In: Proceedings of LICS 1990, pp. 402–413 (1990)
11. Henzinger, T.: Half-order modal logic: How to prove real-time properties. In: Proceedings of the Ninth Annual Symposium on Principles of Distributed Computing, pp. 281–296. ACM Press, New York (1990)
12. Lutz, C.: The complexity of reasoning with concrete domains. PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany (2002)
13. Plaza, J.: Logics of public communications. In: 4th International Symposium on Methodologies for Intelligent Systems, pp. 201–216 (1989)
14. van Benthem, J.: Logics for information update. In: TARK 2001: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 51–67. Morgan Kaufmann Publishers Inc, San Francisco (2001)
15. van Benthem, J., van Eijck, J., Kooi, B.: Logics of communication and change. Information and Computation 204(11), 1620–1662 (2006)
16. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Kluwer academic publishers, Dordrecht (2007)

Canonical Signed Calculi, Non-deterministic Matrices and Cut-Elimination

Arnon Avron and Anna Zamansky

School of Computer Science
Tel-Aviv University, Ramat Aviv, 69978, Israel

Abstract. Canonical propositional Gentzen-type calculi are a natural class of systems which in addition to the standard axioms and structural rules have only logical rules where exactly one occurrence of a connective is introduced and no other connective is mentioned. Cut-elimination in such systems is fully characterized by a syntactic constructive criterion of coherence. In this paper we extend the theory of canonical systems to the considerably more general class of *signed calculi*. We show that the extended criterion of coherence fully characterizes only analytic cut-elimination in such calculi, while for characterizing strong and standard cut-elimination a stronger criterion of density is required. Modular semantics based on non-deterministic matrices are provided for every coherent canonical signed calculus.

1 Introduction

The possibility to eliminate cuts is a crucial property of useful sequent calculi. This property was first established by Gerhard Gentzen in his classical paper "Investigations Into Logical Deduction" ([12]) for sequent calculi for classical and intuitionistic logic. Since then many other cut-elimination[1] theorems, for many systems, have been proved by various methods. Now showing that a given sequent calculus admits cut-elimination is a difficult task, often carried out using heavy syntactic arguments and based on many case-distinctions. It is thus important to have some useful simple criteria that *characterize* cut-elimination (i.e., conditions which are both necessary and sufficient for having an appropriate cut-elimination theorem).

In the same seminal paper ([12]) Gentzen also established an important tradition in the philosophy of logic, according to which the syntactic rules of a proof system determine the semantic meaning of a logical connective in proof systems of an "ideal type". In [2] the idea of such "well-behaved" propositional Gentzen-type systems was formalized by defining "canonical rules and systems" in precise terms. These are systems which in addition to the standard axioms and structural rules have only logical rules where exactly one occurrence of a

[1] We note that by 'cut-elimination' we shall mean in this paper the *existence* of proofs without (certain forms of) cuts, rather than an algorithm to transform a given proof to a cut-free one (the term "cut-admissibility" is sometimes used for cases without non-logical axioms, but this notion is too weak for our purposes).

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 31–45, 2009.

connective is introduced and no other connective is mentioned. In these systems cut-elimination is fully characterized by a constructive syntactic criterion of *coherence*. Moreover, the coherence of a canonical system is equivalent to the existence of a semantic characterization of this system in terms of two-valued *non-deterministic matrices* (Nmatrices), a natural generalization of the standard multi-valued matrices (see e.g. [14]).

In this paper we extend the theory of canonical systems, in particular the characterization of various forms of cut-elimination, to the class of *signed calculi* ([7,10]), of which Gentzen-type systems are particular (two-signed) instances. We show that canonical signed calculi are indeed "well-behaved" in the two senses discussed above. First of all, simple and constructive criteria for characterizing various notions of cut-elimination can be defined for these calculi. Namely, we show that the criterion of *coherence*, extended to the context of signed calculi, fully characterizes only *analytic* cut-elimination in such calculi, while for characterizing strong and standard cut-elimination, a strictly stronger criterion of *density* is required. Secondly, we use finite Nmatrices to provide semantics for canonical signed calculi, and demonstrate that the principle of *modularity* of Nmatrices, which was studied in the context of various non-classical logics (see, e.g. [4]), but never discussed in the context of canonical systems, applies also in this context. We start by providing semantics for the most basic canonical system, and then proceed to show that the semantics of a more complex system is obtained by straightforwardly combining the semantic effects of each of the added rules. As a result, the semantic effect of each syntactic rule taken separately can be analyzed (which is impossible in standard multi-valued matrices). This provides a concrete interpretation of Gentzen's thesis that the meaning of a logical connective is dictated by its introduction rules.

2 Preliminaries

In what follows, $\mathcal{L}$ is a propositional language and $Frm_{\mathcal{L}}$ is the set of wffs of $\mathcal{L}$. $\mathcal{V}$ is a finite set of signs.
Signed calculi ([15,7,10]) manipulate sets of signed formulas, while the signs can be thought of as syntactic markers which keep track of the formulas in the course of a derivation.

Definition 1. *A* signed formula *for $\mathcal{L}$ and $\mathcal{V}$ is an expression of the form $s : \psi$, where $s \in \mathcal{V}$ and ψ is a formula of $\mathcal{L}$. A signed formula $s : \psi$ is atomic if ψ is an atomic formula. A* sequent *is a finite set of signed formulas. A* clause *is a sequent consisting of atomic signed formulas.*

Formulas will be denoted by φ, ψ, signed formulas - by $\alpha, \beta, \gamma, \delta$, sets of signed formulas - by Υ, Λ, sequents - by Ω, Σ, Π, sets of sets of signed formulas - by Φ, Ψ and sets of sequents - by Θ, Ξ. We write $s : \Delta$ instead of $\{s : \psi \mid \psi \in \Delta\}$, $S : \psi$ instead of $\{s : \psi \mid s \in S\}$, and $S : \Delta$ instead of $\{s : \psi \mid s \in S, \psi \in \Delta\}$.

Note 1. The usual (two-sided) sequent notation $\Gamma \Rightarrow \Delta$ can be interpreted as $\{f : \Gamma\} \cup \{t : \Delta\}$, i.e. a sequent in the sense of Definition 1 over $\{t, f\}$.

Definition 2. *Let v be a function from the set of formulas of $\mathcal{L}$ to $\mathcal{V}$.*

1. *v satisfies a signed formula $\gamma = (l : \psi)$, denoted by $v \models (l : \psi)$, if $v(\psi) = l$.*
2. *v satisfies a set of signed formulas Υ, denoted by $v \models \Upsilon$, if there is some $\gamma \in \Upsilon$, such that $v \models \gamma$.*

Thus sequents are interpreted as a *disjunction* of statements, saying that a particular formula takes a particular truth-value (interpreting sequents in a dual way corresponds to the method of analytic tableaux, see e.g. [6,13]).

Non-deterministic matrices are a natural generalization of the notion of a standard multi-valued matrix, in which the value of a complex formula can be chosen non-deterministically out of a non-empty set of options. Below we shortly reproduce the basic definitions from [2].

Definition 3. *A* non-deterministic matrix (Nmatrix) *for $\mathcal{L}$ is a tuple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where:*

- *$\mathcal{V}$ is a non-empty set of truth values (signs).*
- *$\mathcal{D}$ (designated truth values) is a non-empty proper subset of $\mathcal{V}$.*
- *For every n-ary connective $\diamond$ of $\mathcal{L}$, $\mathcal{O}$ includes a corresponding function $\tilde{\diamond} : \mathcal{V}^n \to 2^{\mathcal{V}} \setminus \{\emptyset\}$.*

A valuation $v : Frm_{\mathcal{L}} \to \mathcal{V}$ is legal in an Nmatrix $\mathcal{M}$ *if for every n-ary connective $\diamond$ of $\mathcal{L}$:*

$$v(\diamond(\psi_1, ..., \psi_n)) \in \tilde{\diamond}(v(\psi_1), ..., v(\psi_n))$$

Note that in deterministic matrices the truth-value assigned by a valuation v to a complex formula is uniquely determined by the truth-values of its subformulas. This is not the case in Nmatrices, as v makes a non-deterministic choice out of the set of options $\tilde{\diamond}(v(\psi_1), ..., v(\psi_n))$ and so the semantics defined above is not truth-functional.

Proposition 1. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be an Nmatrix for $\mathcal{L}$ and let v_p be an $\mathcal{M}$-legal partial valuation defined on a set S of $\mathcal{L}$-formulas closed under subformulas (i.e., $\psi_1, ..., \psi_n \in S$ whenever $\diamond(\psi_1, ..., \psi_n) \in S$). Then v_p can be extended to a full $\mathcal{M}$-legal valuation.*

Definition 4. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$[2] be some Nmatrix for $\mathcal{L}$. For a set of sequents Θ and a sequent Ω, $\Theta \vdash_{\mathcal{M}} \Omega$ if for every $\mathcal{M}$-legal valuation v: whenever v satisfies all the sequents in Θ, v also satisfies Ω.*

Definition 5. *For a signed calculus G, we shall write $\Theta \vdash_G \Omega$ whenever a sequent Ω is derivable from a set of sequents Θ in G. We say that $\mathcal{M}$ is a strongly characteristic Nmatrix for G if $\Theta \vdash_G \Omega$ iff $\Theta \vdash_{\mathcal{M}} \Omega$.*

[2] The set of designated truth-values $\mathcal{D}$ in $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ is needed for defining the consequence relation which is induced by $\mathcal{M}$ between *sets of $\mathcal{L}$-formulas* and *$\mathcal{L}$-formulas* (see e.g. [5]). In contrast, the consequence relation $\vdash_{\mathcal{M}}$ used in this paper is between *sets of signed sequents* and *signed sequents*, so the set of designated truth-values plays no role in this context. However, the former consequence relation can be fully characterized in terms of the latter, see e.g. [3].

Note 2. Note that in the case of two-signed calculi (corresponding to Gentzen-type systems, recall Note 1), if $\mathcal{M}$ is strongly characteristic for a (Gentzen-type) system G, then $\mathcal{M}$ is also sound and complete for G in the standard sense.

3 Canonical Signed Calculi

We start by extending the notion of a "Gentzen-type canonical rule" from [2] to the context of signed calculi:

Definition 6. *A* signed canonical rule for a language $\mathcal{L}$ and a finite set of signs $\mathcal{V}$ *is an expression of the form* $[\Theta/S:\diamond]$*, where* S *is a non-empty subset of* $\mathcal{V}$*,* $\diamond$ *is an* n*-ary connective of* $\mathcal{L}$ *and* $\Theta=\{\Sigma_1,...,\Sigma_m\}$*, where* $m\geq 0$ *and for every* $1\leq j\leq m$*:* Σ_j *is a clause consisting of atomic signed formulas of the form* $a:p_k$ *for* $a\in\mathcal{V}$ *and* $1\leq k\leq n$*.*
An application of a signed canonical rule $[\{\Sigma_1,...,\Sigma_m\}/S:\diamond]$ *is an inference step of the following form:*

$$\frac{\Omega,\Sigma_1^* \quad ... \quad \Omega,\Sigma_m^*}{\Omega,S:\diamond(\psi_1,...,\psi_n)}$$

where $\psi_1,...,\psi_n$ *are* $\mathcal{L}$*-formulas,* Ω *is a sequent, and for all* $1\leq i\leq m$*:* Σ_i^* *is obtained from* Σ_i *by replacing* p_j *by* ψ_j *for every* $1\leq j\leq n$*.*

Example 1. 1. The standard Gentzen-style introduction rules for the classical conjunction are usually defined as follows:

$$\frac{\Gamma,\psi,\varphi\Rightarrow\Delta}{\Gamma,\psi\wedge\varphi\Rightarrow\Delta} \qquad \frac{\Gamma\Rightarrow\Delta,\psi \quad \Gamma\Rightarrow\Delta,\varphi}{\Gamma\Rightarrow\Delta,\psi\wedge\varphi}$$

Using the notation from Note 1, we can write $\{f:\Gamma\}\cup\{t:\Delta\}$ (that is, ψ occurs with a sign 'f' if $\psi\in\Gamma$ and with a sign 't' if $\psi\in\Delta$), thus the canonical representation of the rules above is as follows:

$$[\{\{f:p_1,f:p_2\}\}/\{f\}:\wedge] \qquad [\{\{t:p_1\},\{t:p_2\}\}/\{t\}:\wedge]$$

Applications of these rules have the forms:

$$\frac{\Omega\cup\{f:\psi_1,f:\psi_2\}}{\Omega\cup\{f:\psi_1\wedge\psi_2\}} \qquad \frac{\Omega\cup\{t:\psi_1\} \quad \Omega\cup\{t:\psi_2\}}{\Omega\cup\{t:\psi_1\wedge\psi_2\}}$$

2. Consider a calculus over $\mathcal{V}=\{a,b,c\}$ with the following introduction rules for a ternary connective $\circ$:

$$[\{\{a:p_1,c:p_2\},\{a:p_3,b:p_2\}\}/\{a,c\}:\circ(p_1,p_2,p_3)]$$

$$[\{\{c:p_2\},\{a:p_3,b:p_3\},\{c:p_1\}\}/\{b,c\}:\circ(p_1,p_2,p_3)]$$

Their applications are of the forms:

$$\frac{\Omega\cup\{a:\psi_1,c:\psi_2\} \quad \Omega\cup\{a:\psi_3,b:\psi_2\}}{\Omega\cup\{a:\circ(\psi_1,\psi_2,\psi_3),c:\circ(\psi_1,\psi_2,\psi_3)\}}$$

$$\frac{\Omega \cup \{c : \psi_2\} \quad \Omega \cup \{a : \psi_3, b : \psi_3\} \quad \Omega \cup \{c : \psi_1\}}{\Omega \cup \{b : \circ(\psi_1, \psi_2, \psi_3), c : \circ(\psi_1, \psi_2, \psi_3)\}}$$

Definition 7. *Let $\mathcal{V}$ be a finite set of signs.*

1. *A* logical axiom *for $\mathcal{V}$ is a sequent of the form $\{l : \psi \mid l \in \mathcal{V}\}$.*
2. *The cut*[3] *and weakening rules for $\mathcal{V}$ are defined as follows:*

$$\frac{\Omega \cup \{l : \psi \mid l \in L_1\} \quad \Omega \cup \{l : \psi \mid l \in L_2\}}{\Omega \cup \{l : \psi \mid l \in L_1 \cap L_2\}} \textbf{ cut}$$

$$\frac{\Omega}{\Omega, l : \psi} \textbf{ weak}$$

where $L_1, L_2 \subseteq \mathcal{V}$ and $l \in \mathcal{V}$.

The following proposition follows from the completeness of many-valued resolution ([8]):

Proposition 2. *Let Θ be a set of clauses and Ω - a clause. Then Ω follows from Θ iff there is some $\Omega' \subseteq \Omega$, such that Ω' is derivable from Θ by cuts.*

Corollary 1. *For a set of clauses Θ, the empty sequent is derivable from Θ by cuts iff Θ is not satisfiable.*

Proof: Follows from the above proposition and the fact that Θ is unsatisfiable iff the empty sequent follows from Θ.

We are now ready to define the notion of a "canonical signed calculus":

Definition 8. *We say that a signed calculus over a language $\mathcal{L}$ and a finite set of signs $\mathcal{V}$ is* canonical *if it consists of:*

1. *All logical axioms for $\mathcal{V}$.*
2. *The rules of cut and weakening (see Defn. 7).*
3. *A finite number of signed canonical inference rules.*

Although we can define arbitrary canonical signed systems, our main quest is for systems, the syntactic rules of which determine the semantic meaning of the logical connectives they introduce. Thus we are interested in calculi with a "reasonable" (or "non-contradictory") set of rules, which allows for defining a sound and complete semantics for the system (we shall later see that this is also strongly related to cut-elimination). This can be captured by the following simple syntactic *coherence* criterion (which is a generalization of the criterion in [2] for canonical Gentzen-type systems).

Definition 9. *We say that a canonical calculus G is* coherent *if $\Theta_1 \cup ... \cup \Theta_m$ is unsatisfiable whenever $[\Theta_1/S_1 : \diamond], ..., [\Theta_m/S_m : \diamond]$ is a set of rules of G, such that $S_1 \cap ... \cap S_m = \emptyset$.*

[3] The cut is a variation of the basic resolution rule of [8].

Note 3. Obviously, coherence is a decidable property of canonical calculi. We also observe that by Proposition 1, a canonical calculus G is coherent if whenever $\{[\Theta_1/S_1 : \diamond], ..., [\Theta_m/S_m : \diamond]\}$ is a set of rules of G, and $S_1 \cap ... \cap S_m = \emptyset$, we have that $\Theta_1 \cup ... \cup \Theta_m$ is inconsistent (i.e. the empty sequent can be derived from it using cuts). Moreover, we will shortly see that it is not sufficient to check only pairs of rules in the definition above, as it can be the case that $S_1 \cap S_2 \neq \emptyset$ and $S_2 \cap S_3 \neq \emptyset$, but $S_1 \cap S_2 \cap S_3 = \emptyset$.

Example 2. 1. Consider the canonical calculus G_1 over $\mathcal{L} = \{\wedge\}$ and $\mathcal{V} = \{t, f\}$, the canonical rules of which are the two rules for $\wedge$ from Example 1. We can derive the empty sequent from $\{\{t : p_1\}, \{t : p_2\}, \{f : p_1, f : p_2\}\}$ as follows:

$$\dfrac{\dfrac{\{t : p_1\} \quad \{f : p_1, f : p_2\}}{\{f : p_2\}}\ \textbf{cut} \qquad \{t : p_2\}}{\emptyset}\ \textbf{cut}$$

Thus G_1 is coherent.

2. Consider the canonical calculus G_2 over $\mathcal{V} = \{a, b, c\}$ with the following introduction rules for the ternary connective $\circ$:

$$[\{\{a : p_1\}, \{b : p_2\}\}/\{a, b\} : \circ(p_1, p_2, p_3)]$$

$$[\{\{a : p_2, c : p_3\}\}/\{c\} : \circ(p_1, p_2, p_3)]$$

Clearly, the set $\{\{a : p_1\}, \{b : p_2\}, \{a : p_2, c : p_3\}\}$ is satisfiable, thus G_2 is not coherent.

Next we define some notions of cut-elimination in canonical calculi:

Definition 10. Let G be a canonical signed calculus and let Θ be some set of sequents.

1. A cut is called a *Θ-cut* if the cut formula occurs in Θ. We say that a proof is Θ-cut-free if the only cuts in it are Θ-cuts.
2. A cut is called *Θ-analytic* if the cut formula is a subformula of some formula occurring in Θ. A proof is called Θ-analytic[4] if all cuts in it are Θ-analytic. We say that a sequent Ω has a *proper proof from Θ in G* whenever Ω has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G.
3. A canonical calculus G admits *(standard) cut-elimination* if whenever $\vdash_G \Omega$, Ω has a cut-free proof in G. G admits *strong cut-elimination*[5] if whenever $\Theta \vdash_G \Omega$, Ω has in G a Θ-cut-free proof from Θ.
4. G admits *strong analytic cut-elimination* if whenever $\Theta \vdash_G \Omega$, Ω has in G a $\Theta \cup \{\Omega\}$-analytic proof from Θ. G admits *analytic cut-elimination* if whenever $\vdash_G \Omega$, Ω has in G a $\{\Omega\}$-analytic proof.

[4] This is a generalization of the notion of analytic cut (see e.g. [11]).
[5] Strong cut-elimination was characterized for Gentzen-type systems in [5].

Example 3. Consider the following calculus G' for a language with a binary connective $\circ$ and $\mathcal{V} = \{a, b, c\}$. The rules of G' are as follows:

$$R_1 = [\{\{a : p_1\}\}/\{a, b\} : \circ] \quad R_2 = [\{\{a : p_1\}\}/\{b, c\} : \circ]$$

The following proof of $\{b : p_1, c : p_1, b : (p_1 \circ p_2)\}$ in G' is proper, as the cut in the final step is analytic:

$$\dfrac{\dfrac{\{a : p_1, b : p_1, c : p_1\}}{\{b : p_1, c : p_1, b : (p_1 \circ p_2), c : (p_1 \circ p_2)\}}\, R_2 \quad \dfrac{\{a : p_1, b : p_1, c : p_1\}}{\{b : p_1, c : p_1, a : (p_1 \circ p_2), b : (p_1 \circ p_2)\}}\, R_1}{\{b : p_1, c : p_1, b : (p_1 \circ p_2)\}}\, \mathbf{cut}$$

4 Modular Semantics for Canonical Calculi

We now describe a general method of providing modular semantics for canonical signed calculi based on Nmatrices. But first let us explain the intuition behind the need for non-determinism. Consider the standard Gentzen-type introduction rules for negation, which can be represented as follows in terms of signed calculi:

$$[\{t : p_1\}/\{f\} : \neg] \quad [\{f : p_1\}/\{t\} : \neg]$$

The corresponding semantics is of course the classical truth-table for negation, according to which $\tilde{\neg}(t) = f$ (corresponding to the first rule) and $\tilde{\neg}(f) = t$ (corresponding to the second rule). Now suppose we would like to follow intuitionistic logic and discard the second rule, which corresponds to the law of excluded middle. We have a case of *underspecification*, as it is unclear what should now be the truth value of $\tilde{\neg}(f)$. Nmatrices deal with underspecification in a natural way: if $\tilde{\neg}(f)$ is underspecified, then it can be either 't' or 'f', and so we set $\tilde{\neg}(f) = \{t, f\}$.

We start by defining semantics for the most basic signed canonical calculus - the one without *any* canonical rules.

Definition 11. *$G_0^{(\mathcal{L},\mathcal{V})}$ is the canonical calculus for a language $\mathcal{L}$ and a finite set of signs $\mathcal{V}$, whose set of canonical rules is empty.*

Henceforth we assume that our language $\mathcal{L}$ and the set of signs $\mathcal{V}$ are fixed, and so we shall write G_0 instead of $G_0^{(\mathcal{L},\mathcal{V})}$. It is easy to see that G_0 is (trivially) coherent.

We now define a strongly characteristic Nmatrix for G_0. It has the maximal degree of non-determinism in interpreting all of the connectives of $\mathcal{L}$.

Definition 12. *Let $\mathcal{M}_0 = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be the Nmatrix for $\mathcal{L}$, in which for every n-ary connective $\diamond$ of $\mathcal{L}$: $\tilde{\diamond}(a_1, ..., a_n) = \mathcal{V}$ for every $a_1, ..., a_n \in \mathcal{V}$.*

Theorem 1. *$\mathcal{M}_0$ is a strongly characteristic Nmatrix for G_0.*

Proof: The proof is a simplified version of the proof of Theorem 2 in the sequel.

Now we turn to the modular effects of canonical rules: each rule added to the basic canonical calculus imposes a certain semantic condition on the basic Nmatrix $\mathcal{M}_0$, and coherence guarantees that these semantic conditions are not contradictory. For formalizing this we shall need the following technical propositions:

Definition 13. *Let $\mathcal{V}$ be a set of signs. For $\langle a_1, ..., a_n\rangle \in \mathcal{V}^n$, the set of clauses $\mathsf{C}_{\langle a_1,...,a_n\rangle}$ is defined as follows:*

$$\mathsf{C}_{\langle a_1,...,a_n\rangle} = \{\{a_1 : p_1\}, \{a_2 : p_2\}, ..., \{a_n : p_n\}\}$$

Definition 14. We say that a set of clauses Θ is n-canonical if the only atomic formulas occurring in Θ are in $\{p_1, ..., p_n\}$.

Corollary 2. *Let $\Theta_1, \Theta_2..., \Theta_m$ be some n-canonical clauses. If the sets of clauses $\mathsf{C}_{\langle a_1,...,a_n\rangle} \cup \Theta_1, ..., \mathsf{C}_{\langle a_1,...,a_n\rangle} \cup \Theta_m$ are satisfiable, then so is the set $\Theta_1 \cup \Theta_2 ... \cup \Theta_m$.*

Corollary 3. *Let Θ be an n-canonical clause. $\Theta \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent iff for every $\Omega \in \Theta$ there is some $1 \leq i \leq n$, such that $a_i : p_i \in \Omega$.*

Now we define the semantic condition corresponding to each canonical rule:

Definition 15. *Let R be a canonical rule of the form $[\Theta/S : \diamond]$.* **Cond(R)***, the refining condition induced by R is defined as follows:*

$$\textbf{Cond(R)}\text{: For } a_1, ..., a_n \in \mathcal{V}\text{: if } \mathsf{C}_{\langle a_1,...,a_n\rangle} \cup \Theta \text{ is consistent, then } \tilde{\diamond}(a_1, ..., a_n) \subseteq S.$$

Intuitively, a rule $[\Theta/S : \diamond]$ causes the deletion of all the truth-values which are not in S. Whenever some rules $[\Theta_1/S_1 : \diamond], ..., [\Theta_m/S_2 : \diamond]$ "overlap", their overall effect leads to $S_1 \cap ... \cap S_m$ (as we will see below, the coherence of a calculus guarantees that $S_1 \cap ... \cap S_m$ is not empty in such a case).

Definition 16. *Let G be a signed canonical calculus.*

1. *Define an application of a rule $[\Theta/S : \diamond]$ of G for some n-ary connective $\diamond$ on $\overrightarrow{a} = \langle a_1, ..., a_n\rangle \in \mathcal{V}^n$ as follows:*

$$[\Theta/S : \diamond](\overrightarrow{a}) = \begin{cases} S & \text{if } \Theta \cup \mathsf{C}_{\overrightarrow{a}} \text{ is consistent} \\ \mathcal{V} & \text{otherwise} \end{cases}$$

2. *$\mathcal{M}_G = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}\rangle$ is the Nmatrix, in which for every n-ary connective $\diamond$ for $\mathcal{L}$ and every $\overrightarrow{a} = \langle a_1, ..., a_n\rangle \in \mathcal{V}^n$:*

$$\tilde{\diamond}_{\mathcal{M}_G}(\overrightarrow{a}) = \bigcap \{[\Theta/S : \diamond](\overrightarrow{a}) \mid [\Theta/S : \diamond] \in G\}$$

Proposition 3. *If G is coherent, then $\mathcal{M}_G$ is well-defined.*

Proof: It suffices to check that for every n-ary connective $\diamond$ and every $a_1, ..., a_n \in \mathcal{V}$, $\tilde{\diamond}_{\mathcal{M}_G}(a_1, ..., a_n)$ is not empty. Suppose by contradiction that for some n-ary connective $\diamond$ and some $a_1, ..., a_n \in \mathcal{V}$, $\tilde{\diamond}(a_1, ..., a_n) = \emptyset$. But then there are some rules $[\Theta_1/S_1 : \diamond], ..., [\Theta_1/S_m : \diamond]$, such that $S_1 \cap ... \cap S_m = \emptyset$ and $\Theta_1 \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}, ..., \Theta_m \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ are consistent. By Corollary 2, $\Theta_1 \cup ...\Theta_m \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent, and so is $\Theta_1 \cup ... \cup \Theta_m$, in contradiction to our assumption about the coherence of G.

Example 4. Consider a calculus G with the following canonical rules for a unary connective $\bullet$ for $\mathcal{V} = \{t, f, \top, \bot\}$:

$$[\{t : p_1\}/\{t\} : \bullet] \quad [\{f : p_1\}/\{f, \bot\} : \bullet]$$

$$[\{f : p_1, \bot : p_1\}/\{t, \bot\} : \bullet]$$

and the rule for conjunction from Example 1:

$$[\{\{f : p_1, f : p_2\}\}/\{f\} : \wedge]$$

Then the interpretations of $\wedge$ and $\bullet$ in $\mathcal{M}_G$ are as follows:

$\wedge$	t	f	$\top$	$\bot$
t	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\mathcal{V}$
f	$\{f\}$	$\{f\}$	$\{f\}$	$\{f\}$
$\top$	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\mathcal{V}$
$\bot$	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\mathcal{V}$

$\bullet$	
t	$\{t\}$
f	$\{\bot\}$
$\top$	$\mathcal{V}$
$\bot$	$\{t, \bot\}$

Let us explain how the truth-tables are obtained. We start with the basic Nmatrix $\mathcal{M}_0$, for which $\tilde{\bullet}_{\mathcal{M}_0}(x) = \mathcal{V}$ and $\tilde{\wedge}_{\mathcal{M}_0}(x, y) = \mathcal{V}$ for every $x, y \in \mathcal{V}$. Consider the first rule for $\bullet$. Since $\{\{t : p_1\}\}$ is only consistent with $\mathsf{C}_{\langle t\rangle}$, this rule affects $\tilde{\bullet}_{\mathcal{M}_G}(t)$ by deleting the truth-values $f, \top, \bot$ from $\tilde{\bullet}_{\mathcal{M}_0}(t)$, and so $\tilde{\bullet}_{\mathcal{M}_G}(t) = \{t\}$. The second and the third rules both affect the set $\tilde{\bullet}_{\mathcal{M}_G}(f)$ (since the sets $\{\{f : p_1\}\}$ and $\{\{f : p_1, \bot : p_1\}\}$ are both consistent with $\mathsf{C}_{\langle f\rangle}$): the second rule deletes the truth-values $t, \top$, while the third deletes $\top, f$ from $\tilde{\bullet}_{\mathcal{M}_0}$. Thus we are left with $\tilde{\bullet}_{\mathcal{M}_G}(f) = \{\bot\}$. The third rule also dictates $\tilde{\bullet}_{\mathcal{M}_G}(\bot) = \{t, \bot\}$. Finally, as we have underspecification concerning $\tilde{\bullet}_{\mathcal{M}_G}(\top)$, in this case $\tilde{\bullet}_{\mathcal{M}_G}(\top) = \{t, f, \top, \bot\}$. As for the the rule for $\wedge$, the set $\{\{f : p_1, f : p_2\}\}$ is consistent with $\mathsf{C}_{\langle x,y\rangle}$ whenever at least one of $x, y \in \mathcal{V}$ is 'f', and so the rule deletes $t, \top, \bot$ from $\tilde{\wedge}_{\mathcal{M}_0}(x, y)$ for every such x, y.

Now suppose that we obtain a new calculus G' by adding the following rule for $\wedge$ to G (clearly, G' is still coherent):

$$[\{\{t : p_1, \top : p_1\}, \{\bot : p_2, f : p_2\}\}/\{f, \bot\} : \wedge]$$

This rule deletes the truth-values $t, \top$ from $\tilde{\wedge}_{\mathcal{M}_G}(x, y)$ for every $x \in \{t, \top\}$ and $y \in \{f, \bot\}$. Thus the truth-table for $\wedge$ in $\mathcal{M}_{G'}$ is now modified as follows:

$\wedge$	t	f	$\top$	$\bot$
t	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\{f, \bot\}$
f	$\{f\}$	$\{f\}$	$\{f\}$	$\{f\}$
$\top$	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\{f, \bot\}$
$\bot$	$\mathcal{V}$	$\{f\}$	$\mathcal{V}$	$\mathcal{V}$

Note 4. It is easy to see that for a coherent calculus G, $\mathcal{M}_G$ is the weakest refinement of $\mathcal{M}_0$, in which all the conditions induced by the rules of G are satisfied. Thus if G' is a coherent calculus obtained from G by adding a new canonical rule, $\mathcal{M}'_G$ can be straightforwardly obtained from $\mathcal{M}_G$ by some deletions of options as dictated by the condition corresponding to the new rule.

Note 5. It is easy to verify that for the two-sided case studied in [2], the Nmatrix $\mathcal{M}_G$ defined above is similar to the two-valued Nmatrix constructed there. However, our construction of $\mathcal{M}_G$ above is much simpler: a canonical calculus in [2] is first transformed into an equivalent normal form calculus, which is then used to construct the characteristic Nmatrix. The idea is to transform the calculus so that each rule dictates the interpretation for only one tuple $\langle a_1, ..., a_n \rangle$. However, the above definition shows that the transformation into normal form is not necessary and $\mathcal{M}_G$ can be constructed directly from G.

Theorem 2. *For every coherent canonical calculus G, $\mathcal{M}_G$ is a strongly characteristic Nmatrix for G.*

Proof: The proof of strong soundness is not hard and is left to the reader. For strong completeness, suppose that Ω has no proper proof from Θ in G. We will show that this implies $\Theta \nvdash_{\mathcal{M}_G} \Omega$. It is a standard matter to show that Ω can be extended to a maximal set Ω^*, such that (i) no $\Omega' \subseteq \Omega^*$ has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G, and (ii) all formulas occurring in Ω^* are subformulas of formulas from $\Theta \cup \{\Omega\}$. We now show that Ω^* has the following properties:

1. If $\tilde{\diamond}(a_1, ..., a_n) = \{b_1, ..., b_k\}$ and it also holds that $b_1 : \diamond(\psi_1, ..., \psi_n), ..., b_k : \diamond(\psi_1, ..., \psi_n) \in \Omega^*$, then $a_i : \psi_i \in \Omega^*$ for some $1 \leq i \leq n$.
2. For every formula ψ which is a subformula of some formula from Θ, there is exactly one $l \in \mathcal{V}$, such that $l : \psi \notin \Omega^*$.

Let us prove the first property. Suppose by contradiction that for some $a_1, ..., a_n \in \mathcal{V}$, $\tilde{\diamond}(a_1, ..., a_n) = \{b_1, ..., b_k\}$ and $b_1 : \diamond(\psi_1, ..., \psi_n), ..., b_k : \diamond(\psi_1, ..., \psi_n) \in \Omega^*$, but for every $1 \leq i \leq n$, $a_i : \psi_i \notin \Omega^*$. By the maximality of Ω^*, for every $1 \leq i \leq n$ there is some $\Omega'_i \subseteq \Omega^*$, such that $\Omega'_i \cup \{a_i : \psi_i\}$ has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G. First observe that $\{b_1, ..., b_k\} \neq \mathcal{V}$ (otherwise Ω^* would contain a logical axiom, in contradiction to property (i) of Ω^*). Then by definition of $\mathcal{M}_G$ there are some rules in G of the form $R_1 = [\Xi_1 / S_1 : \diamond], ..., R_m = [\Xi_m / S_m : \diamond]$, such that $\Xi_1 \cup \mathsf{C}_{\langle a_1, ..., a_n \rangle}, ..., \Xi_m \cup \mathsf{C}_{\langle a_1, ..., a_n \rangle}$ are consistent and $S_1 \cap ... \cap S_m = \{b_1, ..., b_k\}$. Now let $1 \leq j \leq m$ and $\Sigma \in \Xi_j$. By Corollary 3, there is

some $1 \leq k_\Sigma \leq n$, such that $(a_{k_\Sigma} : p_{k_\Sigma}) \in \Sigma$ (since $\Xi_j \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent). Now by our assumption, $\Omega'_{k_\Sigma} \cup \{a_{k_\Sigma} : \psi_{k_\Sigma}\}$ has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G. By applying weakening we get a $\Theta \cup \{\Omega\}$-analytic proof of $\Omega'_{k_\Sigma} \cup \Sigma^*$ from Θ in G for every $\Sigma \in \Xi_j$, where Σ^* is obtained from Σ by replacing p_r by ψ_r for all $1 \leq r \leq n$. By applying weakening and the canonical rule R_j, we get a $\Theta \cup \{\Omega\}$-analytic proof of $\bigcup_{\Sigma\in\Xi_j} \Omega'_{k_\Sigma} \cup S_j : \diamond(\psi_1, ..., \psi_n)$ from Θ in G. Thus for all $1 \leq j \leq n$, there is some $\Omega_j \subseteq \Omega$, such that $\Omega_j \cup \{S_j : \diamond(\psi_1, ..., \psi_n)\}$ has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G. Now by applying $\Theta \cup \{\Omega\}$-analytic cuts (recall that we assumed that $\{b_1, ..., b_k\} : \diamond(\psi_1, ..., \psi_n) \in \Omega^*$ and so $\diamond(\psi_1, ..., \psi_n)$ is a subformula of some formula from $\Theta \cup \{\Omega\}$), we get a $\Theta \cup \{\Omega\}$-analytic proof of $\Omega_1 \cup ... \cup \Omega_m \cup \{S_1 \cap ... \cap S_m : \diamond(\psi_1, ..., \psi_n)\} = \Omega_1 \cup ... \cup \Omega_m$ from Θ in G , in contradiction to property (i) of Ω^*.

Now we prove the second property. Let ψ be a subformula of some formula from Θ. Then there must be some $l \in \mathcal{V}$, such that $l : \psi \notin \Omega^*$ (otherwise Ω^* contains a logical axiom). Suppose by contradiction that there are some $l_1 \neq l_2$, such that $l_1 : \psi, l_2 : \psi \notin \Omega^*$. By the maximality of Ω^*, then there are some $\Omega'_1, \Omega'_2 \subseteq \Omega^*$, such that $\Omega'_1 \cup \{l_1 : \psi\}$ and $\Omega'_2 \cup \{l_2 : \psi\}$ have $\Theta \cup \{\Omega\}$-analytic proofs from Θ in G. But then by applying ($\Theta \cup \{\Omega\}$-analytic) cuts, we get a $\Theta \cup \{\Omega\}$-analytic proof of $\Omega'_1 \cup \Omega'_2 \subseteq \Omega^*$ from Θ in G, in contradiction to property (i) of Ω^*.

Next we define the partial valuation v on the subformulas of $\Theta \cup \{\Omega\}$ as follows by induction on the complexity of formulas. According to our goal, v is defined so that $v(\psi) \neq s$ for every $(s : \psi) \in \Omega^*$. First, let p be an atomic formula. As Ω^* cannot contain a logical axiom, there must be some $s_0 \in \mathcal{V}$, such that $(s_0 : p) \notin \Omega^*$. Define $v(p) = s_0$. Suppose we have defined v for formulas with complexity up to l, and let $\psi = \diamond(\psi_1, ..., \psi_n)$, where each ψ_i is of complexity at most l. Hence $v(\psi_i)$ is already defined for each i. Now suppose that for every $1 \leq i \leq n$: $v(\psi_i) = a_i$ and $\tilde{\diamond}(a_1, ..., a_n) = \{b_1, ..., b_k\}$. Then there must be some $b \in \{b_1, ..., b_k\}$, such that $(b : \psi) \notin \Omega^*$ (otherwise by property 1 there would be some j, such that $(a_j : \psi_j) \in \Omega^*$, contradicting the induction hypothesis). Define $v(\psi) = b$. By the above construction, v is $\mathcal{M}_G$-legal and $v \not\models_{\mathcal{M}_G} \Omega^*$. Now let $\Sigma \in \Theta$. Then there must be some $a : \psi \in \Sigma$, such that $a : \psi \notin \Omega^*$ (otherwise $\Sigma \subseteq \Omega^*$, while Σ has a $\Theta \cup \{\Omega\}$-analytic proof from Θ in G). By property 2, for every $l \in \mathcal{V} \setminus \{a\}$, $(l : \psi) \in \Omega^*$. By the property of v proven above, $v(\psi) \neq l$ for every $\mathcal{V} \setminus \{a\}$. Thus $v(\psi) = a$, and so $v \models_{\mathcal{M}_G} \Sigma$. By Proposition 1, the partial valuation v can be extended to a full $\mathcal{M}_G$-legal valuation v_f. Thus we have constructed an $\mathcal{M}_G$-legal valuation v_f, such that $v_f \models_{\mathcal{M}_G} \Theta$, but $v_f \not\models_{\mathcal{M}_G} \Omega$. Hence, $\Theta \not\vdash_{\mathcal{M}_G} \Omega$.

From the proof of Theorem 2 we also have the following corollary:

Corollary 4. *Any coherent canonical calculus admits strong analytic cut-elimination.*

Note 6. [3] provides a full axiomatization of finite Nmatrices: a canonical coherent signed calculus is constructed there for every finite Nmatrix. Theorem 2 provides the complementary link between canonical calculi and Nmatrices: every canonical coherent signed calculus has a corresponding finite Nmatrix.

5 Characterization of Cut-Elimination

In this section we provide a characterization of the notions of cut-elimination from Defn. 10. We start with the following theorem, which establishes an *exact correspondence* between coherence of canonical calculi, non-deterministic matrices and strong analytic cut-elimination:

Theorem 3. *Let G be a canonical calculus. The following statements concerning G are equivalent.*

1. *G is coherent.*
2. *G has a strongly characteristic Nmatrix.*
3. *G admits strong analytic cut-elimination.*
4. *G admits analytic cut-elimination.*

Proof: (1) $\Rightarrow$ (2) by Theorem 2.
(1) $\Rightarrow$ (3) by Corollary 4.
(3) $\Rightarrow$ (4) by definition of strong analytic cut-elimination (Defn. 10).
Next we prove that (2) $\Rightarrow$ (1). Suppose that G has a strongly characteristic Nmatrix $\mathcal{M}$ and suppose for contradiction that G is not coherent. Then there are some rules $R_1 = [\Theta_1 : /S_1 : \diamond], ..., R_m = [\Theta_m : /S_m : \diamond]$ in G, such that $\Theta = \Theta_1 \cup ... \cup \Theta_m$ is consistent and $S_1 \cap ... \cap S_m = \emptyset$. By applying the rule R_j on Θ_j for all $1 \leq j \leq m$, we get a proof of $S_j : \diamond(p_1, ..., p_n)$. Then by applying cuts we derive the empty sequent from $\Theta_1 \cup ... \cup \Theta_m$. Let v be any $\mathcal{M}$-legal valuation which satisfies Θ (such valuation exists since Θ is consistent). But by the strong soundness of $\mathcal{M}$ for G, v must then satisfy the empty set, reaching a contradiction.

Finally, we prove that (4) $\Rightarrow$ (1). Suppose that G admits analytic cut-elimination but is not coherent. Then again there are some rules $[\Theta_1 : /S_1 : \diamond], ..., [\Theta_m : /S_m : \diamond]$ in G, such that $\Theta = \Theta_1 \cup ... \cup \Theta_m$ is consistent and $S_1 \cap ... S_m = \emptyset$. Let v be a valuation which satisfies Θ (such valuation exists since Θ is consistent). Let Π be the set of all signed formulas $a : p_i$ (for $1 \leq i \leq n$), such that $v(p_i) \neq a$. Then for every $\Omega \in \Theta$: $\Pi \cup \Omega$ is a logical axiom (indeed, since v satisfies Ω there is some $1 \leq j \leq n$, such that $v(p_j) : p_j \in \Omega$. Then for every $a \in \mathcal{V} \setminus \{v(p_j)\}$, $a : p_j \in \Pi$). By applying the above rules and then cuts Π is provable in G:

$$\dfrac{\dfrac{\Pi \cup \Omega^1_1 \quad ... \quad \Pi \cup \Omega^1_{k_1}}{\Pi \cup S_1 : \diamond(p_1, ..., p_n)} \quad ... \quad \dfrac{\Pi \cup \Omega^m_1 \quad ... \quad \Pi \cup \Omega^m_{k_m}}{\Pi \cup S_m : \diamond(p_1, ..., p_n)}}{\Pi}\ \textbf{cut}$$

where for all $1 \leq j \leq m$: $\Theta_j = \{\Omega^j_1, ..., \Omega^j_{k_j}\}$. Π consists of atomic formulas only and does not contain a logical axiom, and so it has no $\Theta \cup \{\Omega\}$-analytic proof in G (from $\emptyset$), in contradiction to our assumption that G admits analytic cut-elimination.

Next we turn to strong cut-elimination. The following example shows that the quadruple correspondence from Theorem 3 fails if one wants to eliminate also analytic cuts:

Example 5. Consider the calculus G' from Example 3. Clearly, G' is coherent. An analytic proof of the sequent $\{b : p_1, c : p_1, b : (p_1 \circ p_2)\}$ is given in that example. However, it is easy to show that this sequent has no cut-free proof in G'. We will shortly see that by adding the rule $R_3 = [\{\{a : p_1\}\}/\{b\} : \circ]$ to G' cut-elimination is guaranteed.

Thus coherence is not a sufficient condition for strong cut-elimination. Therefore a stronger condition is provided in the next definition:

Definition 17. *A canonical calculus G is* dense *if for every $a_1, ..., a_n \in \mathcal{V}$ and every two rules of G $[\Theta_1/S_1 : \diamond]$ and $[\Theta_2/S_2 : \diamond]$, such that $\Theta_1 \cup \Theta_2 \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent, there is some rule $[\Theta/S : \diamond]$ in G, such that $\Theta \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent and $S \subseteq S_1 \cap S_2$.*

Note 7. It is easy to see that the density of a canonical calculus is decidable (recall also the analogous Note 3 on coherence).

Lemma 1. *Let G be a dense canonical calculus. Let $[\Theta_1/S_1 : \diamond], ..., [\Theta_m/S_m : \diamond]$ be such rules in G, that $\Theta_1 \cup ... \cup \Theta_m$ is consistent. Then for every $a_1, ..., a_n \in \mathcal{V}$, for which $\Theta_1, ..., \Theta_m \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent, there is some $S \subseteq S_1 \cap ... \cap S_m$, such that $[\Theta/S : \diamond]$ is a rule in G and $\Theta \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent.*

Corollary 5. *Every dense canonical calculus is coherent.*

Proof: Suppose that G is dense and let $[\Theta_1/S_1 : \diamond], ..., [\Theta_m/S_m : \diamond]$ be such rules of G, that $S_1 \cap ... \cap S_m = \emptyset$. Suppose by contradiction that $\Theta_1 \cup ... \cup \Theta_m$ is consistent. By Lemma 1, there is some canonical rule $[\Theta/S : \diamond]$ in G, such that $S \subseteq S_1 \cap ... \cap S_m$. By definition of a canonical rule (recall Defn. 6) S is non-empty, in contradiction to our assumption. Thus G is coherent.

Now we can provide an exact characterization of canonical systems which admit standard and strong cut-elimination:

Theorem 4. *Let G be a canonical calculus. The following statements concerning G are equivalent:*

1. *G is dense.*
2. *G admits cut-elimination.*
3. *G admits strong cut-elimination.*

To prove the theorem, first we need the following propositions:

Proposition 4. *Let G be a dense calculus. If Ω has no cut-free proof from Θ in G, then $\Theta \nvdash_{\mathcal{M}_G} \Omega$.*

Proof: Similar to the method used in the proof of Theorem 2.

Lemma 2. *Let G be a coherent calculus over a language with an n-ary connective $\diamond$. Assume that G has at least one canonical rule $[\Theta/S_0 : \diamond]$, such that $\Theta \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent. Let Ω be a set of signed formulas of the form $a : \psi$, where $\psi \in \{p_1, ..., p_n, \diamond(p_1, ..., p_n)\}$. If Ω has a cut-free proof in G, then either (i) Ω is a logical axiom, or (ii) $\diamond(p_1, ..., p_n) \in \Omega$ and there is a rule $[\Xi/S : \diamond]$ in G, such that $\Xi \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent and $S \subseteq \{a \mid a : \diamond(p_1, ..., p_n) \in \Omega\}$.*

Proof of Theorem 4:
$(1 \Rightarrow 3)$: Let G be a dense calculus. Then by Proposition 5, it is also coherent and so $\mathcal{M}_G$ is well-defined. If $\Theta \vdash_G \Omega$, then $\Theta \vdash_{\mathcal{M}_G} \Omega$. Thus by Proposition 4, Ω has a cut-free proof from Θ. Clearly, also $(3 \Rightarrow 2)$ holds. It remains to show that $(2 \Rightarrow 1)$. Suppose that G admits cut-elimination and assume by contradiction that G is not dense. Then there are some $a_1, ..., a_n \in \mathcal{V}$ and some rules $R_1 = [\Theta_1/S_1 : \diamond]$ and $R_2 = [\Theta_2/S_2 : \diamond]$, such that $\Theta_1 \cup \Theta_2 \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent and $S_1 \cap S_2 \neq \emptyset$, but there is no rule $[\Theta/S : \diamond]$ in G, such that $\Theta \cup \mathsf{C}_{\langle a_1,...,a_n\rangle}$ is consistent and $S \subseteq S_1 \cap S_2$. Now let $\Omega_0 = \bigcup_{1\leq i\leq n}\{\mathcal{V} \setminus \{a_i\} : p_i\}$. By lemma 3, for every $\Omega \in \Theta_1 \cup \Theta_2$, there is some $1 \leq i \leq n$, such that $a_i : p_i \in \Omega$. Thus for every $\Omega \in \Theta_1 \cup \Theta_2$, $\Omega \cup \Omega_0$ is a logical axiom. Let $\Theta_1 = \{\Omega_1^1, ..., \Omega_k^1\}$ and $\Theta_2 = \{\Omega_1^2, ..., \Omega_m^2\}$. By applying the two canonical rules and then cuts we get a proof of $\Omega_0 \cup S_1 \cap S_2 : \diamond(p_1, ..., p_n)$ in G:

$$\dfrac{\dfrac{\Omega_1^1 \cup \Omega_0 \quad ... \quad \Omega_k^1 \cup \Omega_0}{\Omega_0 \cup S_1 : \diamond(p_1, ..., p_n)}\, R_1 \qquad \dfrac{\Omega_1^2 \cup \Omega_0 \quad ... \quad \Omega_m^2 \cup \Omega_0}{\Omega_0 \cup S_2 : \diamond(p_1, ..., p_n)}\, R_2}{\Omega_0 \cup (S_1 \cap S_2) : \diamond(p_1, ..., p_n)}\ \mathbf{cut}$$

However, since $\Omega_0 \cup (S_1 \cap S_2) : \diamond(p_1, ..., p_n)$ is not a logical axiom, by Lemma 2 it has no cut-free proof in G, in contradiction to our assumption.

Note 8 (A constructive proof). The semantic proof of cut-elimination in Theorem 4 is not constructive, i.e. it does not provide an algorithm for eliminating cuts in a derivation. A syntactic constructive proof can be obtained by an adaptation of the proof of Theorem 4.1 of [7] to the context of canonical calculi.

Finally, we turn to the special case of canonical calculi with two signs (this includes the canonical Gentzen-type calculi of [2]) the following proposition can be easily shown:

Proposition 5. *A canonical calculus with two signs is dense iff it is coherent.*

Corollary 6. *The following statements concerning a two signed canonical calculus G are equivalent*[6]*:*

1. *G is coherent.*
2. *G is dense.*
3. *G has a strongly characteristic Nmatrix.*
4. *G admits strong analytic cut-elimination.*
5. *G admits analytic cut-elimination.*
6. *G admits strong cut-elimination.*
7. *G admits standard cut-elimination.*

Acknowledgement

This research was supported by the *Israel Science Foundation* founded by the Israel Academy of Sciences and Humanities (grant No 809/06).

[6] This is a generalization of the triple correspondence from Theorem 4.7 of [2] to strong, analytic and strong analytic cut-elimination, which were not handled there.

References

1. Avron, A.: Gentzen-Type Systems, Resolution and Tableaux. Journal of Automated Reasoning 10, 265–281 (1993)
2. Avron, A., Lev, I.: Non-deterministic Multi-valued Structures. Journal of Logic and Computation 15, 241–261 (2005)
3. Avron, A., Konikowska, B.: Proof Systems for Logics Based on Non-deterministic Multiple-valued Structures. Logic Journal of the IGPL 13, 365–387 (2005)
4. Avron, A.: Logical Non-determinism as a Tool for Logical Modularity: An Introduction. In: Artemov, S., Barringer, H., d'Avila Garcez, A.S., Lamb, L.C., Woods, J. (eds.) We Will Show Them: Essays in Honour of Dov Gabbay, vol. 1, pp. 105–124. College Publications (2005)
5. Avron, A., Zamansky, A.: Canonical calculi with (n,k)-ary quantifiers. Journal of Logical Methods in Computer Science 10.2168/LMCS-4(3:2) (2008)
6. Baaz, M., Fermüller, C.G., Salzer, G., Zach, R.: Dual systems of sequents and tableaux for many-valued logics. Bull. EATCS 51, 192–197 (1993)
7. Baaz, M., Fermüller, C.G., Zach, R.: Elimination of cuts in first-order many-valued logics. Journal of Information Processing and Cybernetics 29, 333–355 (1994)
8. Baaz, M., Fermüller, C.G.: Resolution-based theorem proving for many-valued logics. Journal of Symbolic Computation 19(4), 353–391 (1995)
9. Baaz, M., Fermüller, C.G., Zach, R.: Proof theory of finite-valued logics. Bulletin of Symbolic Logic 1 (1995)
10. Baaz, M., Fermüller, C.G., Salzer, G., Zach, R.: Labeled Calculi and Finite-valued Logics. Studia Logica 61, 7–33 (1998)
11. Baaz, M., Egly, U., Leitsch, A.: Normal Form Transformations. Handbook of Automated Reasoning, pp. 273–333 (2001)
12. Gentzen, G.: Investigations into Logical Deduction. In: Szabo, M.E. (ed.) The collected works of Gerhard Gentzen, pp. 68–131. North Holland, Amsterdam (1969)
13. Hähnle, R.: Tableaux for Multiple-valued Logics. In: D'Agostino, M., Gabbay, D., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 529–580. Kluwer Publishing Company, Dordrecht (1999)
14. Urquhart, A.: Many-valued Logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 2, pp. 249–295. Kluwer Academic Publishers, Dordrecht (2001)
15. Rousseau, G.: Sequents in many-valued logic 1. Fundamenta Mathematicae LX, 23–33 (1967)

Temporalization of Probabilistic Propositional Logic

Pedro Baltazar and Paulo Mateus*

SQIG-Instituto de Telecomunicações and IST, Portugal
{pbtz,pmat}@math.ist.utl.pt

Abstract. In this paper we study several properties of the Exogenous Probabilistic Propositional Logic (EPPL), a logic for reasoning about probabilities, with the purpose of introducing a temporal version - Exogenous Probabilistic Linear Temporal Logic (EPLTL). In detail, we give a small model theorem for EPPL and introduce a satisfaction and a model checking algorithm for both EPPL and EPLTL. We are also able to provide a (weakly) complete calculus for EPLTL. Finally, we conclude by pointing out some future work.

1 Introduction

Reasoning about probabilistic systems has been a very important research subject with applications in many fields such as security, performance analysis, system verification, traffic analysis and even bioinformatics. In this context, the use of formal methods, and in particular of logic, via syntactic (computer-aided proof systems) and semantic (model-checking tools) approaches, has been highly beneficial to the community.

In this paper we consider a temporalization of the Exogenous Probabilistic Propositional Logic (EPPL) [16] to reason about the evolution of probability distributions described by probabilistic programs and processes. The term exogenous was coined by Kozen in [11] to express that the probabilities had proper syntax and were not hidden in the propositional symbols or connectives (like in PCTL [1]). The state logic is an extension of the probabilistic logic proposed by Fagin et al [8] where we allow to make classical restrictions over probabilistic spaces. EPPL was initially introduced in [15] to reason about quantum states and further developed in the context of a Hoare-like logic [5]. EPPL semantics is obtained by taking the exogenous semantics approach to enrich a given logic–the models of the enriched logic are sets of models of the given logic with additional structure. This approach was inspired by the possible worlds semantics originally proposed by Kripke [12] for modal logic. A model of EPPL is a

* This work was partially supported by FCT and EU FEDER through PTDC, namely via QSec PTDC/EIA/67661/2006 Project, by Instituto de Telecomunicações via project QuantTel and by the European Network of Excellence - Euro-NF. Pedro Baltazar was also supported by FCT and EU FEDER PhD fellowship SFRH/BD/22698/2005.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 46–60, 2009.

set of possible valuations over propositional symbols (which, for instance, may denote memory cells of a probabilistic program) along with a probability space that gives the probability of each possible valuation. Indeed, as discussed in this paper, EPPL models can be reformulated more precisely as Bernoulli stochastic processes where the index space is the set of propositional symbols.

EPPL differs significantly from probabilistic *arithmetical* assertion logics, such as the state logic of the probabilistic dynamic logic given in [11], where formulas are interpreted as measurable functions and the connectives are arithmetical operations such as addition and subtraction. Inspired by the dynamic logic in [11], there are several important works in probabilistic Hoare logics, *e.g.* [10,17], where the state formulas are either measurable functions or arithmetical formulas interpreted as measurable functions. Intuitively, the Hoare triple $\{f\}\, s\, \{g\}$ means that the expected value of the function g after the execution of s is at least as much as the expected value of the function f before the execution. Although research in probabilistic logics with arithmetical state logics has yielded several interesting results, the formulas themselves do not seem very intuitive. Indeed, a high degree of sophistication is required to write down assertions needed to verify relatively simple programs. For this reason, it is worthwhile to investigate dynamic versions of truth-functional probabilistic logics, such as EPPL. In this paper we present in detail a linear temporalization of EPPL, that we call Exogenous Probabilistic Linear Temporal Logic EPLTL.

The contributions of this paper, taking into account the results presented in [16] are significant. We show that we are able to adapt the technique by [8] to obtain a small model theorem with polynomial bound. Capitalizing in the small model theorem we are able to set a PSPACE bound to the SAT problem for EPPL, which was previously thought to be in EXPSPACE [15]. From the SAT algorithm we are able to derive a simpler Hilbert calculus for EPPL than that presented in [16]. We also discuss in details the model-checking of the logic. Moreover, we are able to provide a complete calculus for the temporal extension, EPLTL, together with a SAT and model-checking algorithm.

This paper is structured as follows. In Section 2 we present the main results concerning EPPL. In Section 3 we present the linear temporalization of EPPL, and in Section 4 we point out some future directions.

2 Probabilistic State Logic

2.1 Syntax

Following the exogenous approach, the language of EPPL consists of formulas at two levels. The formulas of the first level – *basic formulae* – allow us to reason about program variables and states, that at this point we abstract as a finite set of propositional symbols Λ. The formulas of the second level – *global formulae* – allow us to perform probabilistic reason. We also consider *probabilistic terms*, build over a set of real logic variables Z, to denote real numbers used for quantitative reasoning at the global formulae level. The syntax of the language is given by mutual recursion as presented in Table 1.

Table 1. EPPL syntax

$\beta := p \,[\!]\, (\neg\beta) \,[\!]\, (\beta \Rightarrow \beta)$	basic formulae
$t := z \,[\!]\, 0 \,[\!]\, 1 \,[\!]\, (\int\beta) \,[\!]\, (t+t) \,[\!]\, (t \cdot t)$	probabilistic terms
$\delta := (\Box\beta) \,[\!]\, (\beta \perp\!\!\!\perp \beta) \,[\!]\, (t \leq t) \,[\!]\, (\sim\delta) \,[\!]\, (\delta \supset \delta)$	global formulae

where $p \in \Lambda$, $z \in Z$.

Concerning basic formulae, ranged over by $\beta, \beta_1, \ldots$, we assume the usual propositional abbreviations for falsum $\bot$, disjunction $(\beta_1 \vee \beta_2)$, conjunction $(\beta_1 \wedge \beta_2)$ and equivalence $(\beta_1 \Leftrightarrow \beta_2)$.

The probability terms, ranged over by $t, t_1, \ldots$, denote the real numbers. We assume a finite set of (deterministic) real variables, Z, ranging over algebraic real numbers. The probability terms also contain the 0 and 1 real constants that, together with addition, multiplication and the set of logical variables, allow us to express all algebraic real numbers [2]. The probability term $(\int\beta)$ denotes the probability of the set of elements that satisfy β. The terms of the kind $(\int\beta)$ shall henceforth be called *measure terms*.

The global formulas, ranged over by $\delta, \delta_1, \ldots$, are built from modal formulas $(\Box\beta)$, independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$, comparison formulas $(t_1 \leq t_2)$ and the connectives $\sim, \supset$. The modal formula $(\Box\beta)$ allows us to impose restrictions on the probability space, namely to impose that all elements of the sample space satisfy β. We shall also use $(\Diamond\beta)$ as an abbreviation for $(\sim(\Box(\neg\beta)))$. Intuitively, $(\Diamond\beta)$ is satisfied if there is at least one valuation in the probability measure that satisfies β. Observe that $\Box$ and $\Diamond$ are not full fledged modalities, since they cannot be nested[1]. The independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ states that the event described by β_1 is independent from the event β_2.

Other global connectives $\{\mathbf{f}, \cup, \cap, \equiv\}$ and comparison predicates $\{=, \neq, \geq, <, >\}$ are introduced as abbreviations in the classical way. For instance, the global falsum $\mathbf{f}$ stands for $(\Box p \cap (\sim\Box p))$ and $(t_1 = t_2)$ stands for $((t_1 \leq t_2) \cap (t_2 \leq t_1))$.

The notion of occurrence of a term t and a global formula δ_1 in the global formula δ is defined as usual. The same holds for the notion of replacing zero or more occurrences of probability terms and global formulas. For the sake of clarity, we shall often drop parentheses in formulas and terms if it does not lead to ambiguity.

We shall also identify here a useful sublanguage of probabilistic state formulas which do not contain any occurrence of a measure term.

$$\kappa := (\alpha \leq \alpha) \,[\!]\, (\sim\kappa) \,[\!]\, (\kappa \supset \kappa)$$
$$\alpha := z \,[\!]\, 0 \,[\!]\, 1 \,[\!]\, (\alpha + \alpha) \,[\!]\, (\alpha.\alpha)$$

The terms of this sublanguage will be called *analytical terms* and the formulas will be called *analytical formulas*. This language is relevant because it is possible to apply the SAT algorithm for the existential theory of the real numbers to any analytical formula.

[1] We do not have formulas such as $\Box(\Box\beta)$.

2.2 Semantics

The models of EPPL are tuples $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ where $(\Omega, \mathcal{F}, \mu)$ is a probability space and $\mathbf{X} = (X_p)_{p \in \Lambda}$ is a *stochastic process* over $(\Omega, \mathcal{F}, \mu)$ where each X_p is a Bernoulli random variable, that is, X_p ranges over $\mathbf{2} = \{0, 1\}$. Observe that each basic EPPL formula β induces a Bernoulli random variable $X_\beta : \Omega \to \mathbf{2}$ defined as follows: $X_{(\neg\beta)}(\omega) = 1 - X_\beta(\omega)$; and $X_{(\beta_1 \Rightarrow \beta_2)}(\omega) = \max((1 - X_{\beta_1}(\omega)), X_{\beta_2}(\omega))$. So, each basic formula β will represent the measurable subset $\{\omega \in \Omega : X_\beta(\omega) = 1\}$. Moreover, each $\omega \in \Omega$ induces a valuation v_ω over Λ such that $v_\omega(p) = X_p(\omega)$, for all $p \in \Lambda$. Given an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and attribution $\rho : Z \to \mathbb{R}$ for real variables, the denotation of probabilistic terms is as follows:

- $[\![z]\!]_{m,\rho} = \rho(z)$; $[\![0]\!]_{m,\rho} = 0$; $[\![1]\!]_{m,\rho} = 1$;
- $[\![t_1 + t_2]\!]_{m,\rho} = [\![t_1]\!]_{m,\rho} + [\![t_2]\!]_{m,\rho}$; $[\![t_1.t_2]\!]_{m,\rho} = [\![t_1]\!]_{m,\rho}.[\![t_2]\!]_{m,\rho}$; and
- $[\![(\int \beta)]\!]_{m,\rho} = \int X_\beta \, d\mu = \mu(X_\beta^{-1}(1))$

Note that the term $[\![(\int \beta)]\!]_{m,\rho}$ gives the expected value of X_β. Since X_β is a Bernoulli random variable, the expected value is the same as the probability of observing an outcome ω, such that v_ω satisfies β.

Moreover, the satisfaction of global formulas is given by: $m, \rho \Vdash (\Box\beta)$ iff $\Omega = X_\beta^{-1}(1)$; $m, \rho \Vdash (\beta_1 \perp\!\!\!\perp \beta_2)$ iff $X_{\beta_1} \perp\!\!\!\perp X_{\beta_2}$; $m, \rho \Vdash (t_1 \leq t_2)$ iff $[\![t_1]\!]_{m,\rho} \leq [\![t_2]\!]_{m,\rho}$; $m, \rho \Vdash (\sim\delta)$ iff $m, \rho \nVdash \delta$; and $m, \rho \Vdash (\delta_1 \supset \delta_2)$ iff $m, \rho \Vdash \delta_2$ or $m, \rho \nVdash \delta_1$.

Probabilistic terms without occurrences of real variables are called *closed terms*. A global formula only involving closed terms is called a *closed global formula*. Clearly, the denotation of closed terms is independent of the attribution. Consequently, the satisfaction of closed global formulas are also independent of the attribution. So, in these cases, we drop the attribution from the notation.

Remark 1. To design a SAT algorithm for EPPL it is important to make some observations on EPPL models. Let $V_m = \{v_\omega : \omega \in \Omega\}$ be the set of all valuations over Λ induced by m. The basic cylinders*, also called* rectangles*, of an EPPL model m are the subsets $B(b_1 \dots b_k) = \{v \in V_m : v(p_1) = b_1, \dots, v(p_k) = b_k\}$ for $k \geq 0$, $p_1, \dots, p_k \in \Lambda$ and $b_1, \dots, b_k \in \mathbf{2}$. Let $\mathcal{B}_m$ be the set of all basic cylinders of m. Observe that an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ induces a probability space $P_m = (V_m, \mathcal{F}_m, \mu_m)$ over valuations, where $\mathcal{F}_m \subseteq \mathbf{2}^{V_m}$ is the σ-algebra generated by the basic cylinders $\mathcal{B}_m$ and μ_m is defined over basic cylinders by $\mu_m(B) = \mu(\{\omega \in \Omega : v_\omega \in B\})$ for all $B \in \mathcal{B}_m$. Moreover, given a probability space over valuations, $P = (V, \mathcal{F}, \mu)$ we can construct an EPPL model $m_P = (V, \mathcal{F}, \mu, \mathbf{X})$ where $X_p(v) = v(p)$. It is easy to see that m and m_{P_m} satisfy precisely the same formulas. This means that it is enough for a SAT algorithm to search for probability spaces over valuations.*

Given that we are working towards a complete Hilbert calculus for EPPL through a SAT algorithm, it is relevant to understand whether EPPL fulfills a small model theorem. If this is the case then an upper bound on the size of the satisfying models would imply the decidability of the logic (since it would be enough to search for models up to this bound).

Remark 2. The semantic of the independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ allow us to substitute in global formulas all its occurrences by the conjunction

$$((\int \beta_1 \wedge \beta_2) \leq (\int \beta_1)(\int \beta_2)) \bigcap ((\int \beta_1)(\int \beta_2) \leq (\int \beta_1 \wedge \beta_2)).$$

2.3 Small Model Theorem

To obtain a small model theorem we start by defining a quotient construction. Let δ be an EPPL formula. We denote the sets of inequalities and basic subformulas occurring in δ by $iq(\delta)$ and $bsf(\delta)$, respectively. Moreover, we denote the (finite) set of propositional symbols that appear in δ by $prop(\delta)$. Given a formula δ and an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$, we define the following relation on the sample space Ω: $\omega_1 \sim_\delta \omega_2$ iff $X_p(\omega_1) = X_p(\omega_2)$ for all $p \in prop(\delta)$.

Let $prop_\omega(\delta)$ be the subset of propositional symbols of δ such that $X_p(\omega) = 1$. We denote by $[\omega]_\delta$ the $\sim_\delta$ class of ω. Taking an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and an EPPL formula δ, we define the *quotient model* $m/\sim_\delta = (\Omega', \mathcal{F}', \mu', \mathbf{X}')$ where: $\Omega' = \Omega/\sim_\delta$ is the finite set of $\sim_\delta$ classes; $\mathcal{F}' = 2^{\Omega'}$ is the power set σ-algebra; $\mu'(B) = \mu(\cup B)$ for all $B \in \mathcal{F}'$; and $X'_p([\omega]_\delta) = X_p(\omega)$ for all $p \in \Lambda$.

Next, we check that the quotient model is well defined.

Proposition 1. Let $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ be an EPPL model and δ an EPPL formula, then $m/\sim_\delta = (\Omega', \mathcal{F}', \mu', \mathbf{X}')$ is a finite EPPL model .

Now, we prove that satisfaction is preserved by the quotient construction and, consequentially, that any satisfiable formula has a finite discrete EPPL model of size bounded by the formula length. We take the *length* of a basic formula, probabilistic terms or global formula, to be the number of symbols required to write the formula or term. The length of a formula or term ξ is denoted by $|\xi|$.

We are now able to establish a small model theorem for EPPL. We refer the reader to the Appendix for a detailed proof of the result. Observe that at first sight, to construct a model for a formula δ, we need $O(2^{|\delta|})$ algebraic real numbers to describe the probability measure of the σ-algebra over the propositional symbols occurring in δ. However, adapting a technique for eliminating spurious variables in linear programming (already used in [8]), we are able to set this bound to be just linear.

Theorem 1 (Small Model Theorem). *If δ is a satisfiable EPPL formula then it has a finite model using at most $2|\delta| + 1$ algebraic real numbers.*

The small model theorem does not put a bound on the size of the representation of the algebraic real numbers. Indeed, an algebraic real number can be represented as the root of a polynomial of integers, and this polynomial could increase without any bound. Fortunately, thanks to the fact that the existential theory of the real numbers can be decided in PSPACE [4], we find a bound on the size of the real representations in function of the size of the formula, which will lead to a SAT algorithm for EPPL.

2.4 Decision Algorithm for Satisfaction

The decision algorithm for EPPL satisfaction uses the decidability of the existential theory of the real numbers and the small model theorem. Before presenting the algorithm we introduce some notation. Like before, given an EPPL formula δ we will denote by $iq(\delta)$ the set of all subformulas of δ of the form $(t_1 \leq t_2)$. Moreover, we denote by $bf_{\Box}(\delta)$ the set of all subformulas of δ of the form $\Box\beta$, by $ip(\delta)$ the set of all subformulas of the form $\beta_1 \perp\!\!\!\perp \beta_2$ and by $at(\delta)$ the set of all global atoms of δ, that is, $at(\delta) = bf_{\Box}(\delta) \cup iq(\delta) \cup ip(\delta)$. By an *exhaustive conjunction ε of literals of $at(\delta)$* we mean that ε is of the form $\alpha_1 \cap \ldots \cap \alpha_k$ where each α_i is either a global atom or a negation of a global atom. Moreover, all global atoms or their negation occur in ε, so, $k = |at(\delta)|$. At this stage, we consider the EPPL formula $\hat{\varepsilon}$ where in ε all global atoms $\beta_1 \perp\!\!\!\perp \beta_2$ are substitute by the global conjunction in Remark 2. Given a global formula δ, we denote by $\delta^{\alpha}_{p_\alpha}$ the propositional formula obtained by replacing in δ each global atom α with a fresh propositional symbol p_α, and replacing the global connectives $\sim$ and $\supset$ by the propositional connectives $\neg$ and $\Rightarrow$, respectively. We denote by v_ε the valuation over propositional symbols p_α such that $v_\epsilon(p_\alpha) = 1$ iff α occurs positively in ε.

Given an exhaustive conjunction ε of literals of $at(\delta)$, we denote by $lbf_{\Box}(\varepsilon)$ the set of basic formulas such that $\beta \in lbf_{\Box}(\varepsilon)$ if $\Box\beta$ occurs positively in ε (that is, not negated). Similarly, the set of basic formulas that occur nested by a $\sim\Box$ in ε is denoted by $lbf_{\Diamond\neg}(\varepsilon)$. Finally, we denote all the inequalities occurring in $\hat{\varepsilon}$ by $liq(\varepsilon)$. This last set contains the new inequations introduced by the substitution of the independence formulas.

Given a global formula α in $liq(\varepsilon)$ we denote by $\hat{\alpha}$ the analytical formula where all terms of the form $(\int\beta)$ are replaced in α by $\sum_{v \in V, v \Vdash \beta} x_v$ where each x_v is a fresh variable. We need the PSPACE SAT algorithm of the existential theory of the reals numbers [4], that we denote by *SatReal*. We assume that this algorithm either returns *no model*, if there is no solution for the input system of inequations, or a *solution array* ρ, where $\rho(x)$ is the solution for variable x. We denote by $var(\delta)$ the set of real logical variables that occur in δ. Given a solution ρ for a system with X variables and a subset $Y \subseteq X$, we denote by $\rho_{|Y}$ the function that maps each element y of Y to $\rho(y)$.

Theorem 2. Algorithm 1 decides the satisfiability of an EPPL formula in PSPACE.

2.5 Completeness

In [16] it is shown that a superset of axioms and inference rules in Table 2 is a sound and a (weakly) complete axiomatization of EPPL. Herein, and thanks to the EPPL SAT algorithm, we are able to show that the calculus presented in Table 2 is weakly complete.

It is impossible to obtain a strongly complete axiomatization for EPPL (that is, if $\Delta \Vdash \delta$ then $\Delta \vdash \delta$, for arbitrary large Δ, possibly infinite set) because the

Algorithm 1: SatEPPL(δ)

Input: EPPL formula δ
Output: (V, μ) (denoting the EPPL model $m = (V, 2^V, \mu, \mathbf{X})$) and attribution ρ or *no model*

compute $bf_{\Box}(\delta), ip(\delta), iq(\delta)$ *and* $at(\delta)$;
foreach exhaustive conjunction ε of literals of $at(\delta)$ such that $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$ **do**
  compute $lbf_{\Box}(\varepsilon)$, $lbf_{\Diamond\neg}(\varepsilon)$ and $liq(\varepsilon)$;
  foreach $V \subseteq 2^{prop(\delta)}$ such that $0 < |V| \leq 2|\delta| + 1$, $V \Vdash \wedge lbf_{\Box}(\varepsilon)$ and $V \not\Vdash \beta$ for all $\beta \in lbf_{\Diamond\neg}(\varepsilon)$ **do**
    $\kappa \longleftarrow \left(\sum_{v\in V} x_v = 1\right) \cap \left(\bigcap_{v\in V} 0 \leq x_v\right)$;
    foreach $\alpha \in liq(\varepsilon)$ **do**
      $\kappa \longleftarrow \kappa \cap \hat{\alpha}$;
    end
    $\rho \longleftarrow SatReal(\kappa)$;
    if $\rho \neq$ *no model* **then**
      $\mu_\rho \longleftarrow \rho_{|\{x_v : v\in V\}}$;
      $\rho_\rho \longleftarrow \rho_{|var(\delta)}$;
      return (V, μ_ρ) and attribution ρ_ρ;
    end
  end
end
return (*no model*);

logic is not compact [16]. Nevertheless, weakly completeness is enough for software verification, since a program specification generates a finite number of hypotheses.

Concerning the axiomatization of Table 2, we consider an Hilbert system - recursive set of axioms and finitary rules. We recall the axiom schema **ROF** is decidable thanks to Tarski's result on the decidability of real ordered fields. Thus, the axioms in Table 2 constitute a recursive set. Note that the **ROF** axiom allow us to separate the reasoning about probabilities from the reasoning about real numbers.

We can simplify the proof in [16] thanks to the SAT algorithm presented before. The soundness of the calculus of Table 2 is straightforward, and so, we focus on the completeness result. Again, we refer the reader to the Appendix for a detailed proof.

Theorem 3. *The set of rules and axioms of Table 2 is a weakly complete axiomatization of EPPL.*

2.6 Model Checking

Given a finite set of propositional symbols Λ and using the small model theorem we can consider that all EPPL models are defined over a discrete and finite probability space.

Since for the model-checking procedure we have to deal with computer representation and, in practice, probabilities are represented by floating points

Table 2. HC_{EPPL} : complete calculus for EPPL

Axioms
[**CTaut**] $\vdash_{\mathsf{EPPL}} (\Box\beta)$ for each valid propositional formula β;
[**GTaut**] $\vdash_{\mathsf{EPPL}} \delta$ for each instantiation of a propositional tautology δ;
[**Lift** $\Rightarrow$] $\vdash_{\mathsf{EPPL}} (\Box(\beta_1 \Rightarrow \beta_2) \supset (\Box\beta_1 \supset \Box\beta_2))$;
[**Eqv** $\bot$] $\vdash_{\mathsf{EPPL}} (\Box\bot \equiv \mathbf{f})$;
[**Indep**] $\vdash_{\mathsf{EPPL}} (\beta_1 \perp\!\!\!\perp \beta_2) \equiv ((\int\beta_1 \wedge \beta_2) = (\int\beta_1)(\int\beta_2))$;
[**ROF**] $\vdash_{\mathsf{EPPL}} (t_1 \leq t_2)$ for each instantiation of a valid analytical inequality;
[**Prob**] $\vdash_{\mathsf{EPPL}} ((\int\top) = 1)$;
[**FAdd**] $\vdash_{\mathsf{EPPL}} (((\int(\beta_1 \wedge \beta_2)) = 0) \supset ((\int(\beta_1 \vee \beta_2)) = (\int\beta_1) + (\int\beta_2)))$;
[**Mon**] $\vdash_{\mathsf{EPPL}} (\Box(\beta_1 \Rightarrow \beta_2) \supset ((\int\beta_1) \leq (\int\beta_2)))$;
Inference rules
[**MP**] δ_1, $(\delta_1 \supset \delta_2) \vdash_{\mathsf{EPPL}} \delta_2$.

and not symbolically by algebraic real numbers, we consider only EPPL models $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ specified with floating point arrays (like is usual in other probabilistic model checkers, such as PRISM [14,13]). Observe that, since floating point numbers are rational numbers, they are also algebraic real numbers and so, the semantics given in Section 2.2 does not require any modification to deal with floating points. We represent an EPPL model as a $|\Lambda| \times |\Omega|$-matrix $\mathbf{X}$ of boolean values for the random variables and an $|\Omega|$-array μ of real numbers for the probabilities. The size of Ω is at most $2^{|\Lambda|}$. So, an EPPL models is stored in memory by the record $(\mu, \mathbf{X})$.

Let δ be an EPPL global formula. We consider that in δ we have already replace all occurrences of independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ by inequalities as describe in Remark 2.

We define the arrays $bsf(\delta) = (\beta_1, \ldots, \beta_k)$, $pst(\delta) = (t_1, \ldots, t_s)$ and $gsf(\delta) = (\delta_1, \ldots, \delta_m, \delta)$ as the ordered tuples of basic subformulas, probabilistic subterms and global subformulas of δ, respectively, ordered by increasing length. An attribution ρ for real logical variables is also represented by a finite array where the dimension is determined by the number of real logical variables in the formula $|\delta|$, that is bounded by s (the length of $pst(\delta)$). As usual for floating points, we assume that the basic arithmetical operations take $O(1)$ time.

Given an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$, an attribution ρ and a global formula δ, *the model-checking problem* consists in determining whether $m, \rho \Vdash \delta$. Model checking of EPPL is detailed in Algorithm 2.

Theorem 4. Assuming that all basic arithmetical operations and that accessing array/matrix values take $O(1)$ time, Algorithm 2 takes $O(|\delta| \cdot |\Omega|)$ time to decide if an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and attribution ρ satisfy δ.

3 Probabilistic Linear Time Logic

3.1 Linear Time Logic

Syntax. We assume that there is a countable set of propositional symbols Ξ. Assuming the set Ξ, the formulas of Linear Time Logic (LTL) are given in BNF notation as

Algorithm 2: CheckEPPL(m, ρ, δ)

Input: EPPL model $m = (\mu, \mathbf{X})$, attribution ρ and a formula δ
Output: Boolean value $G(|gsf(\delta)|)$

for $i = 1$ **to** $|bsf(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch β_i **do** /* each case takes $O(|\Omega|)$ time */
    case $p:\ B(i) = X_p$;
    case $(\neg\beta_j):\ B(i) = 1 - B(j)$;
    case $(\beta_j \Rightarrow \beta_l):\ B(i) = \max(1 - B(j), B(l))$;
  end
end
for $i = 1$ **to** $|pst(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch t_i **do** /* each case takes $O(|\Omega|)$ */
    case $z:\ T(i) = \rho(z)$:;
    case 0 **or** 1 : $T(i) = t_i$;
    case $(\int\beta_j):\ T(i) = B(j).\mu$; /* this case takes $O(2|\Omega|)$ */
    case $(t_j + t_l):\ T(i) = T(j) + T(l)$;
    case $(t_j.t_l):\ T(i) = T(j).T(l)$;
  end
end
for $i = 1$ **to** $|gsf(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch δ_i **do** /* each case takes $O(|\Omega|)$ */
    case $(\Box\beta_j):\ G(i) = \Pi_{l=1}^{|\Omega|} B(j,l)$; /* this case takes $O(|\Omega| - 1)$ */
    case $(t_j \leq t_l):\ G(i) = (T(j) \leq T(l))$;
    case $(\delta_j \supset \delta_l):\ G(i) = \max(1 - G(j), G(l))$;
  end
end

$$\theta := \mathbf{f} \,[\!]\, p \,[\!]\, (\theta \supset \theta) \,[\!]\, \mathsf{X}\theta \,[\!]\, \theta\mathsf{U}\theta$$

where $p \in \Xi$.

Semantics. The semantics of the temporal logic LTL is given using a Kripke structure. A *Kripke structure* over a set of propositions Ξ is a tuple $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ where S is a set, elements of which are called *states*; $\mathsf{R} \subseteq S \times S$ is a said to be the *accessibility relation* and it is assumed that for every $s \in \mathsf{S}$ there exists $s' \in \mathsf{S}$ such that $(s, s') \in \mathsf{R}'$; and $\mathsf{L} : \mathsf{S} \to \wp(\Xi)$ is said to be a *labeling function.* Given a Kripke structure, $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$, an infinite sequence of states $\pi = s_1 s_2 \ldots$ is said to be a *computation path* if $(s_i, s_{i+1}) \in \mathsf{R}$ for all $i \geq 1$. The semantics of LTL is defined in terms of a Kripke structure $\mathcal{K}$ and a computation path π. The LTL modalities contain symbols for temporal reasoning: X stands for ne**x**t; and U for **u**ntil. The remaining temporal modalities, F and G, are easily obtained by abbreviation: $(\mathsf{F}\theta)$ for $((\sim\mathbf{f})\mathsf{U}\theta)$; and $(\mathsf{G}\theta)$ for $(\sim\mathsf{F}(\sim\theta))$.

Given a Kripke structure $\mathcal{K}$, a computation path $\pi = s_1 \ldots$ of the Kripke structure, and a LTL formula θ, the semantics is defined inductively in terms of

Table 3. Semantics of LTL

$\mathcal{K}, \pi \not\Vdash_{\mathsf{LTL}} \mathbf{f}$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} p$ iff $p \in \mathsf{L}(s_1)$ with $\pi = s_1, \ldots$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} (\theta_1 \supset \theta_2)$ iff $\mathcal{K}, \pi \not\Vdash_{\mathsf{LTL}} \theta_1$ or $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \theta_2$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \mathsf{X}\theta$ iff $\mathcal{K}, \pi^2 \Vdash_{\mathsf{LTL}} \theta$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} (\theta_1 \mathsf{U} \theta_2)$ iff there is some $i \geq 1$ such that $\mathcal{K}, \pi^i \Vdash_{\mathsf{LTL}} \theta_2$ and $\mathcal{K}, \pi^j \Vdash_{\mathsf{LTL}} \theta_1$ for $1 \leq j < i$.

a relation $\mathcal{K}, \pi \Vdash \theta$ and is given in Table 3. We denote by π^i the i-th suffix of π, that is, the path $s_i, s_{i+1} \ldots$

A Kripke strucute $\mathcal{K}$ is a model of (or satisfies) the formula θ if $\mathcal{K}, \pi \Vdash \theta$ for every path π in $\mathcal{K}$. As usual, we say that a set of formulas Θ entails the formula θ, which we write $\Theta \vDash \theta$, if a Kripke structure satisfying all the formulas in Θ also satisfies θ.

Axiomatization. The temporal logic LTL enjoys a sound and complete axiomatization. The proof system HC_{LTL} of LTL is given in Table 4.

The following result is proved in [9].

Theorem 5. *The proof system HC_{LTL} is sound and weakly complete with respect to Kripke structures.*

3.2 Exogenous Probabilistic Linear Time logic

Syntax. The formulas of Exogenous Probabilistic Linear Time Logic (EPLTL) are obtained by enriching the probabilistic formulas with LTL modalities and are depicted in Table 5. The temporal modalities $\mathsf{F}\theta$ and $\mathsf{G}\theta$ are introduced as abbreviations. Observe that the connectives $\mathbf{f}$ and $\supset$ are shared with EPPL.

Semantics. We now provide a semantics for EPLTL based on EPPL-parametrized Kripke structures. An *EPPL-parametrized Kripke structure* is a tuple $\mathcal{T} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$,

Table 4. HC_{LTL} : complete calculus for LTL

Axioms
[**Taut**] All propositional tautologies with propositional symbols substituted by LTL formulas;
[**X1**] $\vdash_{\mathsf{LTL}} (\sim \mathsf{X}\theta_1) \equiv (\mathsf{X} \sim \theta_1)$
[**X2**] $\vdash_{\mathsf{LTL}} (\mathsf{X}(\theta_1 \supset \theta_2)) \supset (\mathsf{X}\theta_1 \supset \mathsf{X}\theta_2)$
[**G**] $\vdash_{\mathsf{LTL}} (\mathsf{G}\theta_1) \supset (\theta_1 \cap (\mathsf{X}\mathsf{G}\theta_1))$
[**U1**] $\vdash_{\mathsf{LTL}} (\theta_1 \mathsf{U} \theta_2) \supset (\mathsf{F}\theta_2)$
[**U2**] $\vdash_{\mathsf{LTL}} (\theta_1 \mathsf{U} \theta_2) \equiv (\theta_2 \cup (\theta_1 \cap \mathsf{X}(\theta_1 \mathsf{U} \theta_2)))$
Inference rules
[**MP**] $\theta_1, (\theta_1 \supset \theta_2) \vdash_{\mathsf{LTL}} \theta_2$
[**XGen**] $\theta_1 \vdash_{\mathsf{LTL}} (\mathsf{X}\theta_1)$
[**Ind**] $(\theta_1 \supset \theta_2), (\theta_1 \supset (\mathsf{X}\theta_1)) \vdash_{\mathsf{LTL}} (\theta_1 \supset (\mathsf{G}\theta_2))$

Table 5. Language of EPLTL

$\theta := \mathbf{f} \,[\!]\, \gamma \,[\!]\, (\theta \supset \theta) \,[\!]\, (\mathsf{X}\theta) \,[\!]\, (\theta \mathsf{U} \theta)$ where γ is an EPPL formula.

Table 6. Semantics of EPLTL

$\mathcal{T}, \pi \not\Vdash_{\mathsf{EPLTL}} \mathbf{f}$	
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \gamma$	iff $m_1, \rho_1 \Vdash_{\mathsf{EPPL}} \gamma$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\theta_1 \supset \theta_2)$	iff $\mathcal{T}, \pi \not\Vdash_{\mathsf{EPLTL}} \theta_1$ or $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \theta_2$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\mathsf{X}\theta)$	iff $\mathcal{T}, \pi^2 \Vdash_{\mathsf{EPLTL}} \theta$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\theta_1 \mathsf{U} \theta_2)$	iff there is some $i \geq 1$ such that $\mathcal{T}, \pi^i \Vdash_{\mathsf{EPLTL}} \theta_2$ and $\mathcal{T}, \pi^j \Vdash_{\mathsf{EPLTL}} \theta_1$ for $1 \leq j < i$.

where S is a non-empty set of states; $\mathsf{R} \subseteq \mathsf{S} \times \mathsf{S}$ is a total relation; and a function L such that $\mathsf{L}(s)$ is a pair (m, ρ), for each $s \in S$, where m is an EPPL model and ρ is an attribution. Like for Kripke structures, a computation path is a infinite sequence $\pi = m_1\rho_1, m_2\rho_2 \ldots$ such that for any $i \geq 1$, we have $(m_i\rho_i, m_{i+1}\rho_2) \in \mathsf{R}$. Given an EPPL-Kripke structure $\mathcal{T}$, a computational path π in $\mathcal{T}$ and a EPLTL formula θ, the semantics of EPLTL is defined in terms of a relation $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \gamma$ given in Table 6.

An EPPL-Kripke structure $\mathcal{T}$ is said to satisfy an EPLTL formula θ, which we denote by $\mathcal{T} \Vdash_{\mathsf{EPLTL}} \theta$, if $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \theta$ for all computational paths π in $\mathcal{T}$. The entailement relation is defined as for LTL.

3.3 Axiomatization

We are able to provide a weakly complete axiomatization of EPLTL capitalizing on the complete LTL calculus HC_{LTL}, which we present in Table 7. Please note that although the completeness of the calculus may look trivial, the proof of completeness is subtle. This is because the connectives $\mathbf{f}$ and $\supset$ are shared between EPPL and LTL which may create new theorems that would not be obtained by just adding the EPPL axioms to LTL axioms.

It is straightforward to check the soundness of the calculus, for this reason we omit here the lengthy exercise of verifying that all axioms and inference rules are sound.

Theorem 6 (Soundness). *The axiomatization HC_{EPLTL} is sound.*

Table 7. HC_{EPLTL} calculus for EPLTL

Axioms	
[**PTeo**]	All EPPL theorems;
[**LTLTaut**]	All LTL theorems with propositional symbols substituted by EPLTL formulas.
Inference rules	
[**PMP**]	$\theta_1, (\theta_1 \supset \theta_2) \vdash_{\mathsf{QCTL}} \theta_2$;
[**Gen**]	$\theta_1 \vdash_{\mathsf{LTL}} \mathsf{G}\theta_1$.

The completeness of the calculus is established by translating probabilistic atoms into propositional symbols. Consider the subset of atomic EPPL formulas pAtom (*i.e.*, the set constituted by box formula ($\Box\beta$), independence formulas ($\beta_1 \perp\!\!\!\perp \beta_2$) and comparison formulas ($t_1 \leq t_2$)). Let Ξ be the countable set of propositional symbols used to write LTL formulas. Given a fixed bijective map $\lambda : \mathsf{pAtom} \rightarrow \Xi$ (that translates each global atom to a LTL propositional symbol) we can translate each EPPL formula θ to a LTL formula $\lambda(\theta)$ by extending inductively λ on the structure of the formula θ (and preserving all connectives). For simplicity, we denote $\lambda(\theta)$ just by $\widetilde{\theta}$. The map λ can also be used to translate an EPPL-Kripke structure $\mathcal{T} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ to the Kripke structure $\widetilde{\mathcal{T}} = (\mathsf{S}, \mathsf{R}, \widetilde{\mathsf{L}})$, where $p \in \widetilde{\mathsf{L}}(s)$ if $\mathsf{L}(s) \Vdash_{\mathsf{EPPL}} \lambda^{-1}(p)$.

Lemma 1. *Let $\mathcal{T}$ be an EPPL-Kripke structure. Then, $\mathcal{T}, \pi \Vdash_{EPLTL} \theta$ iff $\widetilde{\mathcal{T}}, \pi \Vdash_{LTL} \widetilde{\theta}$.*

The next lemma shows that EPLTL incorporates both LTL and EPPL reasoning.

Lemma 2. *Let θ be and EPLTL formula, if $\vdash_{LTL} \widetilde{\theta}$ then $\vdash_{EPLTL} \theta$. Moreover, let γ be an EPPL formula, if $\vdash_{EPPL} \gamma$ then $\vdash_{EPLTL} \gamma$.*

Proof. Follows directly from axioms **LTLTaut** and **PTeo**.

If one restricts just to EPPL formulas, EPLTL reasoning coincides with that of EPPL.

Lemma 3. *Let γ be an EPPL formula. Then $\vdash_{EPLTL} \gamma$ iff $\vdash_{EPPL} \gamma$.*

The following lemma is crucial to the proof of completeness.

Lemma 4. *Let θ be an EPLTL formula such that $\Vdash_{EPLTL} \theta$. Then there is an EPPL formula γ_θ such that $\vdash_{EPLTL} \gamma_\theta$ and $\Vdash_{LTL} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$.*

We are now able to show the completeness of HC_{EPLTL}.

Theorem 7. *The axiomatization HC_{EPLTL} is weakly complete.*

Proof. Let $\Vdash_{\mathsf{EPLTL}} \theta$ be a valid EPLTL formula. With γ_θ as in Lemma 4 we have that $\Vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$. Using LTL completeness we have $\vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$. Now, from Lemma 2 we get $\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta \supset \theta)$.

Hence, we are able do the following derivation in EPLTL:

1) $\vdash_{\mathsf{EPLTL}} \gamma_\theta$	Lemma 4
2) $\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta)$	Rule **Gen** at 1
3) $\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta \supset \theta)$	Lemma 4,Lemma 2
4) $\vdash_{\mathsf{EPLTL}} \theta$	Rule **PMP** at 2,3

Therefore, HC_{EPLTL} is complete.

3.4 SAT Problem

Let θ be the EPLTL formula that we want to test for satisfiability and $at = \{\gamma_1, \ldots, \gamma_k\}$ be the set of atomic EPPL formulas that are atoms of θ. Now for each k-vector $i \in \{0,1\}^k$, consider the EPPL formula

$$\delta_i = \prod_{j=1}^{k} \varphi_j \qquad \text{where} \qquad \varphi_j = \begin{cases} \gamma_j & \text{if } j\text{-th bit of } i \text{ is } 1 \\ (\sim\gamma_j) & \text{otherwise} \end{cases} \tag{1}$$

Let $K \subseteq \{0,1\}^k$ be such that δ_i is an EPPL consistent formula and let $\gamma_\theta = \bigsqcup_{i \in K} \delta_i$. Observe that $\Vdash_{\mathsf{EPPL}} \gamma_\theta$ and that for each EPPL model m, ρ there exists a unique $i \in K$ such that $m, \rho \Vdash_{\mathsf{EPPL}} \delta_i$. Given a Kripke structure $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ that satisfies $(\mathsf{G}(\widetilde{\gamma_\theta}) \cap \widetilde{\theta})$ and a path π starting at $s \in \mathsf{S}$ we denote by (m_s, ρ_s) an EPPL model that satisfies δ_i whenever $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\delta_i}$. Moreover, choose $(m_s, \rho_s) \neq (m_{s'}, \rho_{s'})$ whenever $s \neq s'$ (this can be done just by changing the assignments of variables not occurring in θ). We denote by $\mathcal{T}_\mathcal{K}$ the EPPL-Kripke structure $(S_\mathcal{K}, R_\mathcal{K}, id : S_\mathcal{K} \to S_\mathcal{K})$ where $S_\mathcal{K} = \{(m_s, \rho_s) : s \in S\}$ and $((m_s, \rho_s), (m_{s'}, \rho_{s'})) \in R_\mathcal{K}$ iff $(s, s') \in R$. Finally, given a computation path $\pi = s_1, \ldots$ in $\mathcal{K}$ we denote by π_K the computation path $(m_{s_1}, \rho_{s_1}), \ldots$ in $\mathcal{T}_\mathcal{K}$.

The following theorem is the kernel of the EPLTL SAT algorithm.

Theorem 8. *Let θ be a EPLTL formula. Then, $(\mathsf{G}(\widetilde{\gamma_\theta}) \cap \widetilde{\theta})$ is LTL-satisfiable iff θ is EPLTL-satisfiable. Moreover, $\mathcal{K}, \pi \Vdash_{LTL} (\mathsf{G}(\widetilde{\gamma_\theta}) \cap \widetilde{\theta})$ iff $\mathcal{T}_\mathcal{K}, \pi_\mathcal{K} \Vdash_{EPLTL} \theta$.*

We are now able to show the SAT algorithm for EPLTL.

Algorithm 3: SAT Algorithm for EPLTL– Sat$\mathsf{EPLTL}(\theta)$

Input: EPLTL formula θ
Output: $\mathcal{T}$ (an EPPL-Kripke structure satisfying θ) or *no model*

compute δ_i *as in Equation (1) for all* $i \in 2^k$;
let $K = \{i \in 2^k : Sat\mathsf{EPPL}(\delta_i) \neq$ "*no model*"$\}$ *and*
  $\mathcal{M} = \{(m_i, \rho_i) : i \in K$ *and* $Sat\mathsf{EPPL}(\delta_i) = (m_i, \rho_i)\}$;
let $\gamma_\theta = \bigsqcup_{i \in K} \delta_i$;
let $\mathcal{K} = Sat\mathsf{LTL}(\mathsf{G}(\widetilde{\gamma_\theta}) \cap \widetilde{\theta})$;
if $\mathcal{K} =$*no model* **then**
  return *(no model)*;
end
return *($\mathcal{T}_\mathcal{K}$ constructed from the models stored in $\mathcal{M}$)*;

The SAT algorithm for EPLTL that we present is EXPSPACE, due to applying the PSPACE SAT algorithm of LTL [18] to a formula that has increased exponentially. It is possible to improve this bound by adapting the LTL SAT algorithm in order to cope with EPPL formulas. In a full version of the paper it is worthwhile presenting this improvement, but for the sake of space, we preferred herein to present a simpler algorithm.

3.5 Model-Checking Problem

Similarly to the SAT algorithm, we provide a model-checking algorithm for EPLTL that uses directly the PSPACE model-checking algorithm for LTL. This makes the problem more or less trivial thanks to Lemma 1. Given the EPLTL formula θ and a EPPL-Kripke structure $\mathcal{T}$, we start by transforming $\mathcal{T}$ into a classical Kripke structure $\widetilde{\mathcal{T}}$ by checking whether the EPPL models in $\mathcal{T}$ satisfy or not the probabilistic atoms in θ. Then, it remains to model check in LTL the Kripke structure $\widetilde{\mathcal{T}}$ against $\widetilde{\theta}$. Clearly, the model-checking procedure is in PSPACE, since the translation of $\mathcal{T}$ into $\widetilde{\mathcal{T}}$ can be done in polynomial space.

4 Future Work

A research line we will pursue is on using SAT solvers for predicate abstraction [7]. We will explore how EPPL can be applied on probabilistic predicate abstraction. Following the work on non-probabilistic verification of C-like programs [6] we also intend to analyze the probabilistic version of the bit-vector logic [3]. We intend to implement and apply the model-checking algorithm to case studies and check the power of the formalism against established temporal probabilistic logics. The temporalization can also be generalized to more rich logics such as the μ-calculus. Another research line to be pursued consists in introducing quantifiers for the real variables that could nest with the temporal modalities. Moreover, it is interesting to understand the relationship between EPPL-parametrized Kripke structures and Markov chains.

Acknowledgements

We would like to thank Christel Baier for helpful comments and suggestions.

References

1. Baier, C., Clarke, E., Hartonas-Garmhausen, V., Kwiatkowska, M., Ryan, M.: Symbolic model checking for probabilistic processes. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 430–440. Springer, Heidelberg (1997)
2. Basu, S., Pollack, R., Marie-Françoise, R.: Algorithms in Real Algebraic Geometry. Springer, Heidelberg (2003)
3. Brinkmann, R., Drechsler, R.: RTL-datapath verification using integer linear programming. In: VLSI Design, pp. 741–746 (2002)
4. Canny, J.: Some algebraic and geometric computations in pspace. In: STOC 1988: Proceedings of the twentieth annual ACM symposium on Theory of computing, pp. 460–469. ACM, New York (1988)
5. Chadha, R., Cruz-Filipe, L., Mateus, P., Sernadas, A.: Reasoning about probabilistic sequential programs. Theoretical Computer Science 379(1-2), 142–165 (2007)
6. Clarke, E.M., Kroening, D., Sharygina, N., Yorav, K.: SATABS: SAT-based predicate abstraction for ANSI-C. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 570–574. Springer, Heidelberg (2005)

7. Clarke, E.M., Talupur, M., Veith, H., Wang, D.: Sat based predicate abstraction for hardware verification. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 78–92. Springer, Heidelberg (2004)
8. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87(1/2), 78–128 (1990)
9. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: The temporal analysis of fairness. In: Proceedings 7th Symp. on Principles of Programming Languages, POPL 1980, pp. 163–173. ACM, New York (1980)
10. Jones, C.: Probabilistic Non-Determinism. PhD thesis, U. Edinburgh (1990)
11. Kozen, D.: A probabilistic PDL. Journal of Computer System Science 30, 162–178 (1985)
12. Kripke, S.A.: Semantical analysis of modal logic. I. Normal modal propositional calculi. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 9, 67–96 (1963)
13. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: Probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002)
14. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic model checking in practice: case studies with PRISM. SIGMETRICS Perform. Eval. Rev. 32(4), 16–21 (2005)
15. Mateus, P., Sernadas, A.: Weakly complete axiomatization of exogenous quantum propositional logic. Information and Computation 204(5), 771–794 (2006)
16. Mateus, P., Sernadas, A., Sernadas, C.: Exogenous semantics approach to enriching logics. In: Sica, G. (ed.) Essays on the Foundations of Mathematics and Logic, Polimetrica, vol. 1, pp. 165–194 (2005)
17. Morgan, C., McIver, A., Seidel, K.: Probabilistic predicate transformers. ACM Transactions on Programming Languages and Systems 18(3), 325–353 (1996)
18. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. J. ACM 32(3), 733–749 (1985)

Logic and Bounded-Width Rational Languages of Posets over Countable Scattered Linear Orderings

Nicolas Bedon

Université Paris-Est and CNRS
Laboratoire d'informatique de l'Institut Gaspard Monge, UMR CNRS 8049
5, boulevard Descartes, Champs-sur-Marne
77454 Marne-la-Vallée Cedex 2
France
Nicolas.Bedon@univ-mlv.fr

Abstract. In this paper we consider languages of labelled N-free posets over countable and scattered linear orderings. We prove that a language of such posets is series-rational if and only if it is recognizable by a finite depth-nilpotent algebra if and only if it is bounded-width and monadic second-order definable. This extends previous results on languages of labelled N-free finite and ω-posets and on languages of labelled countable and scattered linear orderings.

1 Introduction

It is known since the beginning of the 60's that automata, rational expressions, monadic second-order logic (MSO[$<$]) and finite monoids all have the same expressive power for the definition of languages of finite words. They have been widely studied since that time, and adapted to more complex structures, like infinite words, trees, or partially ordered labelled sets. The subject of this paper is such an extension, from two different directions: languages of infinite words, and languages of posets. Let us cite the works it relies on.

First, Büchi [6,7] initiated the study of automata on infinite words, in the meaning that letters are indexed by ordinal numbers, and not just integers as for the finite words case. He successfully used the equivalent expressive power of automata and MSO[$<$] in order to exhibit decision procedures for the decidability of this logic interpreted over ordinals. Extending the automata technique to trees, Rabin [18] studied the decidability of the monadic second-order theory of trees, from which he deduced decidability results on linear orders. Later, Bruyère and Carton [5] introduced automata and equivalent rational expressions for words whose shape is a countable and scattered linear ordering. These automata and rational expressions were recently connected to MSO[$<$] by Bedon, Bès, Carton and Rispal [2]. Adding the notion of parallelism to the notion of sequentiality in words, Lodaya and Weil [14] defined automata, rational expressions and finite algebra for languages of particular finite partially ordered sets, obtained using

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 61–75, 2009.

the letters and closure under sequential and parallel composition. This class of posets corresponds precisely to the class of N-free posets [22,23]. The connection between the results of Lodaya and Weil and MSO[<] has been established by Kuske [11,12], together with an extension to the infinite (ω) composition of N-free finite posets. Finally, Bedon and Rispal [3] recently extended the Kleene-like theorems of Bruyère–Carton and Lodaya–Weil by defining automata and rational expressions for languages of N-free posets with a sequential composition indexed by a countable and scattered linear ordering.

This paper is a study of the connections between MSO[<], finite algebra and rational expressions for this class (denoted by $SP^\diamond$) of posets. We extend to $SP^\diamond$ the results of Kuske on N-free ω-posets and those of Bedon, Bès, Carton and Rispal on countable and scattered linear orderings. We prove that MSO[<], finite depth-nilpotent algebra and series-rational expressions have the same expressive power for bounded-width languages of posets of $SP^\diamond$. The logical formalism is due to Büchi, but interpreted over $SP^\diamond$ rather than well-ordered structures. The series-rational languages are those of Bruyère and Carton, extended with a commutative operation for finite parallel composition. Finally, the algebra are semigroups equipped with a sequential product adapted for sequences of elements indexed by scattered and countable linear orderings, and a commutative parallel composition of elements. Informally speaking, such an algebra is depth-nilpotent if any nested parallel composition leads to 0 beyond a fixed threshold. As the constructions are effective, this gives in particular a decision procedure that relies on automata theory for the theory of MSO[<] interpreted over bounded-width posets of $SP^\diamond$. This decision result can also be obtained without automata theory, using for example model-theoretical techniques as in [20], or the methods of [13]. As a corollary, the inclusion problem of rational languages is also decidable.

The paper is organized as follows. Section 2 is devoted to basic definitions, linear orderings and posets. Sections 3, 4 and 5 respectively introduce series-rational languages, algebra and logic for posets over scattered and countable linear orderings. The main result of the paper is stated in Section 6. It contains in particular sketches of constructions to obtain a finite depth-nilpotent algebra from a logical sentence, a logical sentence from a series-rational expression, and a series-rational expression from a finite depth-nilpotent algebra, which shows the equivalence between the three formalisms. Finally, Section 7 concludes.

2 Basic Definitions

We start by some basic definitions on linear orderings. We refer to [19] for a survey on the subject. Let J be a set equipped with an order $<$. The ordering J is *linear* if all elements are comparable : for any distinct j and k in J, either $j < k$ or $k < j$. For any linear ordering J, we denote by $-J$ the backward linear ordering obtained from the set J with the reverse ordering. A linear ordering J is *dense* if for any j and k in J such that $j < k$, there exists an element i of J such that $j < i < k$. It is *scattered* if it contains no dense sub-ordering. The ordering ω of natural integers is scattered. Ordinals are also scattered orderings.

We denote by $\mathcal{N}$ the sub-class of finite linear orderings, $\mathcal{O}$ the class of countable ordinals and $\mathcal{S}$ the class of countable scattered linear orderings.

Definition 1. *A linear ordering J is* complete *if*

- *every non-empty sub-ordering K of J which is bounded above has a least upper bound in J, and*
- *every non-empty sub-ordering K of J which is bounded below has a greatest lower bound in J.*

Example 1. Let $\alpha = \omega$ and $\beta = -\omega$. The scattered linear ordering $\gamma = \alpha + \beta$ is not complete, as the part of γ composed of the elements of α (resp. β) does not respect the first (resp. last) condition of the definition. However, $\omega + 1 + -\omega$ is complete.

Let $J \in \mathcal{S}$ be a countable and scattered linear ordering. An *interval* K of J is a subset $K \subseteq J$ such that $\forall k_1, k_2 \in K, \forall j \in J$, if $k_1 < j < k_2$ then $j \in K$. A *cut* (K, L) of J is a partition of J into two intervals K and L such that all the elements of K are less than all the elements of L. Thus $J = K \cup L$ and $K \cap L = \emptyset$. The set of all cuts of J is denoted by $\hat{J} = \{(K, L) | K \cup L = J \wedge \forall k \in K, \forall l \in L, k < l\}$. The set $\hat{J}$ is naturally equipped with the ordering $(K_1, L_1) < (K_2, L_2)$ if and only if $K_1 \subset K_2$. This linear ordering can be extended to $J \cup \hat{J}$ by setting $j < (K, L)$ whenever $j \in K$ for any $j \in J$ and $(K, L) \in \hat{J}$, and keeping the orderings on the elements of J and of $\hat{J}$. We set $\hat{J}^* = \hat{J} \setminus \{(\emptyset, J), (J, \emptyset)\}$.

A *poset* $(P, <)$ is a set P partially ordered by $<$. In order to lighten the notation we often denote the poset $(P, <)$ by P. An *antichain* is a subset P' of P such that all elements of P' are incomparable (with $<$). The *width* of P is

$$wd(P) = \sup\{|E| : E \text{ is an antichain of } P\}$$

where sup denotes the least upper bound of the set. In this paper, we restrict to *countable scattered posets* which are thus partially ordered countable sets without any dense sub-ordering. Let $(P, <_P)$ and $(Q, <_Q)$ be two disjoint posets. The *parallel composition* of $(P, <_P)$ and $(Q, <_Q)$ is the poset $(P \cup Q, <)$ where $x < y$ if and only if ($x, y \in P$ and $x <_P y$) or ($x, y \in Q$ and $x <_Q y$). The *sum* (or *sequential composition*) $P + Q$ of P and Q is the poset $(P \cup Q, <)$ such that $x < y$ if and only if one of the following conditions is true:

- $x \in P$, $y \in P$ and $x <_P y$;
- $x \in Q$, $y \in Q$ and $x <_Q y$;
- $x \in P$ and $y \in Q$.

The sum of two posets can be generalized to any linearly ordered sequence of pairwise disjoint posets: if J is a linear ordering and $((P_j, <_j))_{j \in J}$ is a sequence of posets, then $\sum_{j \in J} P_j = (\cup_{j \in J} P_j, <)$ such that $x < y$ if and only if ($x \in P_j$, $y \in P_j$ and $x <_j y$) or ($x \in P_j$ and $y \in P_k$ and $j < k$). The sequence $((P_j, <_j))_{j \in J}$ is called a *J-factorization*, or *factorization* for short, of the poset $\sum_{j \in J} P_j$. The only poset $(\emptyset, <)$ of width 0 is called *empty poset* and is denoted by ϵ.

Definition 2. *The class $SP^\diamond$ of* series-parallel *scattered and countable posets is the smallest class of posets containing the empty poset, the singleton and closed under finite parallel composition and sum indexed by countable scattered linear orderings.*

The class $SP^\diamond$ has a nice characterization in terms of graph properties: $SP^\diamond$ coincides with the class of scattered and countable N-free posets without infinite antichain. A poset P is *N-free* if it does not contain N as a sub-poset, that is if it does not contain elements $p, q, r, s \in P$ such that the ordering relations between those four elements are *precisely* $p < r$, $q < r$ and $q < s$.

Theorem 1. *[3] A poset belongs to $SP^\diamond$ if and only if it is N-free, countable and scattered, and without infinite antichain.*

An *alphabet* is a set whose elements are called *letters*. In this paper, we use only finite alphabet, thus the term "finite" is omitted. A poset *labelled* by A is a poset $(P, <)$ equipped with a *labelling* map $l : P \to A$ which associates a letter to any element of P. The notion of a labelled poset corresponds to the notion of *pomset* in the literature. Also, the finite labelled posets of width 1 correspond to the usual notion of words. In order to shorten the notation, we make no distinction between a poset and a labelled poset. The class of posets of $SP^\diamond$ labelled by A is denoted by $SP^\diamond(A)$. A *language* of $SP^\diamond(A)$ is a subset of $SP^\diamond(A)$. The sequential and parallel composition of posets can naturally be extended to languages. If $L_1, L_2 \subseteq SP^\diamond(A)$, then

$$L_1 \cdot L_2 = \{P \in SP^\diamond(A) : \exists P_1 \in L_1 \ \exists P_2 \in L_2 \text{ such that } P = P_1 + P_2\}$$
$$L_1 \parallel L_2 = \{P \in SP^\diamond(A) : \exists P_1 \in L_1 \ \exists P_2 \in L_2 \text{ such that } P = P_1 \parallel P_2\}$$

Let n be an integer. We denote by $SP^\diamond_{\leq n}(A)$ the set of posets of $SP^\diamond(A)$ of width at most n and by $SP^\diamond_*(A)$ the class $SP^\diamond(A)$ restricted to posets of finite width. A language L has *bounded-width* if there exists an integer n such that L contains only posets of width at most n.

We now focus on the definitions of algebras for the recognition of languages. A semigroup $(S, \cdot)$ is a set S equipped with an associative binary operation $\cdot$ called *product*. A *$\|$-semigroup* [14,15,16] $(S, \cdot, \|)$ is an algebra such that $(S, \cdot)$ is a semigroup and $(S, \|)$ is a commutative semigroup. In ambiguous contexts, the $\cdot$ and $\|$ products are respectively called *sequential* (or *series*) and *parallel*. The *$\diamond$-semigroups* are a generalization of semigroups for the recognition of words indexed by countable and scattered linear orderings (see [8] for more details): a *$\diamond$-semigroup* $(S, \prod)$ is a set equipped with a map $\prod$ (also called *sequential product*) which associates an element of S to any countable and linearly ordered sequence $s = (s_j)_{j \in J}$ (with $J \in \mathcal{S}$) of elements of S, such that $\prod(t) = t$ for any element t of S and $\prod$ is associative (i.e. for any factorization of the sequence s into a sequence of sequences $(t_j)_{j \in J'}$, $\prod(s) = \prod((\prod t_j)_{j \in J'})$). Finally, a *$\| -\diamond$-semigroup* $(S, \prod, \|)$ is an algebra such that $(S, \prod)$ and $(S, \|)$ are respectively a $\diamond-$ and a $\|$-semigroup. Recall that an algebra is finite if it is composed of a finite number of elements. Even if a $\| -\diamond$-semigroup is finite, the description of the product $\prod$ is not finite since the

product of any sequence of countable length must be given. Actually, the sequential product of a finite $\| -\diamond$-semigroup can be described in a finite manner: we refer to [8] for this description in the case of finite $\diamond$-semigroups, which also immediately applies to finite $\| -\diamond$-semigroups. Even if the notion of a $\| -\diamond$-semigroup does not really fit into the general framework of universal algebra, the following notions are self understanding (we refer to [1] for precise definitions in the framework of universal algebra): sub-algebra, term, congruence, quotient, morphism between two algebras of the same type (i.e. two semigroups, or $\|$-semigroups, or $\diamond$-semigroups, or $\| -\diamond$-semigroups), free algebra, terms.

In order to lighten the notation we often denote an algebra by its set of elements: for example, we denote the semigroup $(S, \cdot)$ by S. We denote by S^1 the algebra S if S has an identity for all its operations, $S \cup \{1\}$ otherwise, 1 being an identity for all the operations. We also denote by A^+, $SP(A)$ [14,15,16] and $A^\diamond$ [8] respectively the free semigroup, $\|$-semigroup, and $\diamond$-semigroup over the set A. Let S and T be two algebras of the same type. Then S *divides* T if S is the quotient of a sub-algebra of T. A morphism $\varphi : S \to T$ *recognizes* a subset X of S if $\varphi^{-1}\varphi(X) = X$. We say that T recognizes X if there exists a morphism from S into T recognizing X. A subset X of an algebra S is *recognizable* if there exist a finite algebra T with the same type as S and a morphism $\varphi : S \to T$ that recognizes X.

Proposition 1. *Let A be an alphabet. Then $SP^\diamond(A)$ is the free $\| -\diamond$-semigroup over A.*

Example 2. Let $A = \{a, b\}$ and $L \subseteq SP^\diamond_{\leq 2}(A)$ be the language of non-empty posets P such that P has width at most $\overline{2}$ and each letter a that appears into a parallel part of P is incomparable with a b. Let $S = \{a, b, ab, p, 0, 1\}$ be the finite $\| -\diamond$-semigroup defined by the following $\|$ commutative product: $a \parallel a = ab \parallel a = 0$, $p \parallel x = 0$ for all $x \in S$, $a \parallel b = ab \parallel b = ab \parallel ab = b \parallel b = p$ and the sequential product $\prod$ such that, for any non-empty sequence $(s_j)_{j\in\mathcal{J}}$ $(J \in \mathcal{S} - \{\emptyset\})$ of elements of S,

$$\prod((s_j)_{j\in J}) = \begin{cases} a & \text{if } (s_j)_{j\in J} \text{ contains only } a\text{s} \\ ab & \text{if } (s_j)_{j\in J} \text{ contains at least one } a \text{ and one } b \\ b & \text{if } (s_j)_{j\in J} \text{ contains only } b\text{s} \\ p & \text{if } (s_j)_{j\in J} \text{ contains only } p, a, b, ab, \text{ with at least one } p \end{cases}$$

The elements 1 and 0 are respectively neutral and a zero for both $\prod$ and $\|$. Let $\varphi : SP^\diamond(A) \to S$ be the morphism defined by $\varphi(a) = a$ and $\varphi(b) = b$. Then $L = \varphi^{-1}(\{a, b, ab, p\})$.

3 Series-Rational Languages

Let A be an alphabet. Let L and L' be bounded-width languages of $SP^\diamond(A)$. The following operations are used in order to form the series-rational expressions for bounded-width languages of labelled posets:

$$L^* = \{\sum_{j\in n} P_j | n \in \mathcal{N}, P_j \in L\}$$
$$L^\omega = \{\sum_{j\in \omega} P_j | P_j \in L\}$$
$$L^{-\omega} = \{\sum_{j\in -\omega} P_j | P_j \in L\}$$
$$L^\natural = \{\sum_{j\in \alpha} P_j | \alpha \in \mathcal{O}, \alpha \neq 0, P_j \in L\}$$
$$L^{-\natural} = \{\sum_{j\in -\alpha} P_j | \alpha \in \mathcal{O}, \alpha \neq 0, P_j \in L\}$$
$$L_1 \diamond L_2 = \{\sum_{j\in J\cup \hat{J}^*} P_j |\ J \in \mathcal{S} - \{\emptyset\}, P_j \in L_1 \text{ if } j \in J \text{ and } P_j \in L_2 \text{ if } j \in \hat{J}^*\}$$

The class of *series-rational* languages over an alphabet A is the smallest containing ϵ, $\{a\}$ for all $a \in A$, and closed by finite union, finite sequential and parallel compositions, finite sequential iteration *, ω and $-\omega$-iterations, iteration on ordinals $\natural$ and reverse iteration on ordinals $-\natural$ as well as diamond operator $\diamond$. We denote by $L(e)$ the language described by a series-rational expression e. If $E \subseteq SP^\diamond(A)$, then $E^\diamond$ is an abbreviation for $E \diamond \epsilon$. Note that if $\epsilon \notin E$, then $\epsilon \notin E^\diamond$.

A language is *linear-rational* if it is series-rational without using the $\parallel$ operator. Note that the linear-rational expressions are precisely those of Bruyère and Carton [5] over words on scattered and countable linear orderings. The following Theorem is a reformulation of a result of Carton and Rispal on recognizable languages of words on scattered and countable linear orderings.

Theorem 2. *[8] Let A be an alphabet, and L be a language of $SP^\diamond_{\leq 1}(A)$. Then L is linear-rational iff it is recognizable.*

4 Algebra

We now define a pre-ordering relation $\prec_\parallel$, on the elements of a $\parallel -\diamond$-semigroup S, in order to adapt the notion of the depth of an element of a *sp*-algebra [14] to $\parallel -\diamond$-semigroups. If $s, t \in S$, then $s \prec t$ if and only if there exist $x_1, x_2, x_3, x_4, x_5 \in S^1$ such that $s = x_1(x_2 \parallel (x_3 t x_4))x_5$ and $x_2 \parallel (x_3 t x_4) \neq x_3 t x_4$. The relations $\prec_\parallel$ and $\preceq_\parallel$ are respectively the transitive and reflexive-transitive closures of $\prec$. Observe that if S has a zero 0 or an identity 1 for all its operations, then the last condition of the definition of $\prec$ ensures that $0 \not\prec_\parallel 0$ and $1 \not\prec_\parallel 1$. By construction, $\preceq_\parallel$ is a pre-order on S. A $\parallel -\diamond$-semigroup S is *depth-graded* if, for each $s \in S$, there exists an integer n such that each $\prec_\parallel$-chain with s as least element has length at most n. In a depth-graded algebra, the relation $\prec_\parallel$ is irreflexive, and hence is a strict partial order. The *depth* of s, denoted $dp(s)$, is defined inductively on the elements of a depth-graded $\parallel -\diamond$-semigroup S by

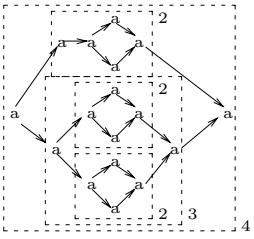


Fig. 1. A finite poset of depth 4. The depth of some sub-posets into frames is indicated.

$$dp(s)=\begin{cases}1 & \text{if there does not exist } t\in S \text{ such that } s\prec_{\|} t,\\ 1+\sup\{dp(t) : s\prec_{\|} t\} & \text{otherwise.}\end{cases}$$

A $\| -\diamond$-semigroup S has *bounded depth* if there exists an integer n such that each $\prec_{\|}$-chain is of length at most n. Observe that if S is bounded-depth, then it is also depth-graded.

Example 3. The poset of Figure 3 has depth 4 and width 6. Observe that $SP^{\diamond}(A)$ is not depth-graded. Consider for example the poset $P=\sum_{i<\omega} P_i$, where P_i consists in i copies of the letter a, all set in parallel, for each $i<\omega$. Then $P\in SP^{\diamond}(A)$, and $dp(P)$ is infinite. Observe that any poset of $SP^{\diamond}_{*}(A)$ has finite depth, and $SP^{\diamond}_{*}(A)$ is depth-graded. However, $SP^{\diamond}_{*}(A)$ does not have bounded-depth.

A $\| -\diamond$-semigroup S is *depth-nilpotent* if it has bounded-depth and $S-\{1\}$ contains a 0 which is the only idempotent for the $\|$ operation. The following Lemma is an immediate adaptation of Lemma 3.5 of [15] to $\| -\diamond$-semigroups:

Lemma 1. *Let S be a $\| -\diamond$-semigroup with bounded-depth. Then S is depth-nilpotent iff for any $s,t\in S-\{1\}$ with $t\neq 0$, then $s\| t\neq t$.*

5 Logic

The monadic second-order (MSO[<]) logic is classical in set theory, and was first set up by Büchi for words. In this paper the formulæ of MSO[<] are interpreted over posets of $SP^{\diamond}(A)$, labelled by the letter of the alphabet A. Formal logic is used to specify a language by properties of its labelled posets, as for example "every antichain X of a poset of the language such that X has at least two elements contains at least two elements labelled by a". We now outline the basic notions on MSO[<]. We refer e.g. to [9,21] for more details.

We first focus on first-order logic. The *first order variables* are named by lower-cases letters like x, y, z and are interpreted over elements of the posets. An existential ($\exists$) or an universal ($\forall$) quantification can be applied to a variable: in this case, the variable is said to be *bounded* to the quantifier. An unbounded variable is *free*. The basic ingredients for formulæ construction are the *atomic*

formulæ. If x, y and a are respectively two first-order variables and a letter of the alphabet, then $R_a(x)$ and $x < y$ are atomic formulæ: the first one meaning that x is associated to an element labelled by a and the other is self-understanding. The logical symbols $\wedge$, $\vee$ and $\neg$ are the usual connectives. Parenthesis ensure legibility of formulæ. We write $\varphi(x_1, \ldots, x_n)$ to denote that φ has at most $x_1, \ldots, x_n$ as free variables. A *sentence* is a formula without free variable.

The satisfaction relation $\models$ between series-parallel labelled posets $(P, <)$ and logic formulæ is defined canonically. Let $\phi(x_1, \ldots, x_n)$ be a formula having at most $x_1, \ldots, x_n$ as free variables, $(P, <)$ be a poset and $p_1, \ldots, p_n \in P$. Then $(P, <), p_1, \ldots, p_n \models \phi(x_1, \ldots, x_n)$ means that $(P, <)$ satisfies φ when $p_1, \ldots, p_n$ serve as respective interpretations for $x_1, \ldots, x_n$.

We define the second-order logic MSO[$<$] as an extension of the first-order logic. In MSO[$<$], variables that range over sets of elements of posets are also allowed in addition to first-order variables. We use upper-cases letters like X, Y, Z to name these variables, called *second-order variables*. Comparatively to first-order logic, MSO[$<$] has one more form of atomic formula, which is self-understanding: $x \in X$, where x and X are respectively a first and a second-order variable. The notions and notations introduced for first-order logic can naturally be extended to second-order logic.

The *language* $L(\phi)$ of a sentence ϕ is the set of all labelled-posets satisfying ϕ. A property p is *definable* in MSO[$<$] if there exists a formula of MSO[$<$] that expresses p.

Let n be an integer, A an alphabet, P and P' two labelled posets. Following the notation from [9], we write $P \equiv_n P'$ if P and P' satisfy the same sentences of quantifier rank $\leq n$ (the quantifier rank is the maximum number of nested quantifiers in a formula). It is a well-known result that $\equiv_n$ is an equivalence relation with finitely many equivalence classes. Furthermore, $SP(A)/\equiv_n$ is a finite $\|$-semigroup (see for example Proposition 3.1.4 of [9]). This is only verification to check, using the same arguments as in the proof of Proposition 3.1.4 of [9], that

Proposition 2. *Let A be an alphabet and n an integer. Then $SP^\diamond(A)/\equiv_n$ (resp. $A^\diamond/\equiv_n$) is a finite $\| -\diamond$-semigroup (resp. $\diamond$-semigroup) that recognizes $L(\phi)$ for any sentence ϕ of quantifier rank $\leq n$.*

In order to enhance readability of formulæ we use several notations and abbreviations for properties expressible in MSO[$<$]. The following are usual and self-understanding: $\phi \to \psi$, $X \subseteq Y$, $x = y$. We denote "there exists an unique x" by $\exists! x$, "x and y are different and not comparable" by $x \| y$, "there exists a non-empty set X" by $\overline{\exists} X$, "set X has cardinality j" by $\texttt{Card}_j(X)$, where j is any integer, "set X is an antichain" by $\texttt{Antichain}(X)$, "sets U and V form a partition of X" by $\texttt{Partition}(U, V, X)$. All those properties are definable in MSO[$<$]. A set X is finite iff every non-empty subset of X has a minimum and a maximum. It is isomorphic to ω (resp. $-\omega$) if it is linearly ordered, infinite and every part of X with a maximum (resp. minimum) is finite. Again, all the properties above are definable in MSO[$<$]. A simple transcription of Definition 1 into MSO[$<$] gives a monadic second-order formula that tests if an ordered set is a complete linear ordering. Finally, we need the following abbreviation:

$$X = U \oplus V \equiv \texttt{Partition}(U, V, X) \wedge (\forall u \forall v \; u \in U \wedge v \in V \rightarrow \neg u \parallel v)$$

Example 4. Let $A = \{a, b\}$ be an alphabet and $L \subseteq SP^{\diamond}(A)$ be the language of posets verifying the property "every antichain X of a poset of the language such that X has at least two elements contains at least two elements labelled by a". Let ϕ be the following sentence of MSO[<]. Then $L(\phi) = L$.

$$\phi \equiv \forall X \; \texttt{Antichain}(X) \rightarrow \exists Y \; Y \subseteq X \wedge \texttt{Card}_2(Y) \rightarrow (\exists x \exists y \; x \in X \wedge y \in X \wedge x \neq y \wedge R_a(x) \wedge R_a(y))$$

6 Equivalence

This section is devoted to the main result of this paper:

Theorem 3. *Let A be an alphabet and $L \subseteq SP^{\diamond}(A)$. Then the following assertions are equivalent:*

1. *L is series-rational,*
2. *there exists a morphism $\varphi : SP^{\diamond}(A) \rightarrow S$ into a finite depth-nilpotent $\parallel -\diamond$-semigroup S and $X \subseteq S$ such that $0 \notin X$, $L = \varphi^{-1}(X)$, and $\varphi(a) \neq 1$ for all $a \in A$,*
3. *L has bounded-width and is definable by a sentence of MSO[<].*

Example 5. Let $A = \{a, b\}$ and L be the language of Example 2. Then L is the language of the series-rational expression

$$e = ((A^{\diamond} + \epsilon)a(A^{\diamond} + \epsilon) \parallel (A^{\diamond} + \epsilon)b(A^{\diamond} + \epsilon) + A^{\diamond} + b^{\diamond} \parallel b^{\diamond})^{\diamond}$$

and of the logical sentence

$$\phi_e \equiv \phi_{\mathrm{wd} \in \{1,2\}} \wedge \forall x (R_a(x) \wedge \neg \texttt{Flat}(x)) \rightarrow \exists y \; x \parallel y \wedge R_b(y)$$

where $\phi_{\mathrm{wd} \in \{1,2\}} \equiv \forall X \texttt{Antichain}(X) \rightarrow \vee_{1 \leq i \leq 2} \texttt{Card}_i(X)$ and $\texttt{Flat}(x) \equiv \forall y \; x = y \vee x < y \vee y < x$. Furthermore, it can be easily checked that the morphism of $\parallel -\diamond$-semigroups of Example 2, that recognizes L, also verifies the properties of Theorem 3.

Observe that the language L of Example 4 is definable in MSO[<], but as it has not bounded width it is not series-rational. It is also recognizable by a morphism $\varphi : SP^{\diamond}(A) \rightarrow S$, with S finite and depth-nilpotent, but it can be easily checked by the reader that in this case $0 \in \varphi(L)$.

In the remainder of this section we will give a sketch of the proof of Theorem 3. Proposition 3 shows that 2 implies 1, and Proposition 4 that 3 implies 2. The proof of Proposition 3 essentially relies on an induction over the depth of $s \in S - \{0\}$. Theorem 2 solves the induction step $dp(s) = 1$. Proposition 4 relies on Proposition 2, whose proof uses classical Ehrenfeucht-Fraïssé games arguments.

Proposition 3. *Let A be an alphabet, S be a finite depth-nilpotent $\| -\diamond$-semigroup and $\varphi : SP^{\diamond}(A) \to S$ a morphism that recognizes $L \subseteq SP^{\diamond}(A)$, such that $0 \notin \varphi(L)$ and $\varphi(a) \neq 1$ for all $a \in A$. Then L is series-rational.*

Proposition 4. *Let A be an alphabet, ϕ a monadic second-order sentence and m an integer. Let $L = L(\phi) \cap SP^{\diamond}_{\leq m}(A)$. There exist a morphism $\varphi : SP^{\diamond}(A) \to S$ into a finite depth-nilpotent $\| -\diamond$-semigroup S and $X \subseteq S$ such that $0 \notin X$, $L = \varphi^{-1}(X)$, and $\varphi(a) \neq 1$ for all $a \in A$.*

As a series-rational language has necessarily bounded-width, it remains to show that it can also be defined using a sentence of MSO[<]. We now give a sketch of the proof, which proceeds by induction on a series-rational expression e, by extension of the ideas from [17]. For simplicity we do not consider the empty poset in the discussion. Let P be a non-trivial poset such that $P \in L(e)$. Then by definition P is obtained from others posets of $SP^{\diamond}(A)$ using the series-rational operators. Let Q be such a non-empty subset of P. Then Q verifies two properties. First, any element of P between two comparable elements of Q also belongs to Q: we say that Q has the *block* property. Second, if an element x of P is comparable with an element of Q and incomparable with another one, then x belongs to Q: we say that Q has the *good part* property. Those two properties are definable in MSO[<]. Let C be a non-empty good part of P which can be decomposed into an union of maximal (with respect to inclusion) blocks. If such a decomposition exists it is unique. We construct, by induction on e, a formula $\varphi_e(X)$ with exactly one free variable X, which is second-order, and such that $P, C \models \varphi_e(X)$ iff $D \in L(e)$ for each maximal block D of C. The basic step of the induction, where e is a letter, is trivial. The induction step where e has the form $e_1 \parallel e_2$ (resp. $e_1 \cdot e_2$) is easy: it suffices to express in MSO[<] that each maximal block D of C can be partitioned into U and V such that all the elements of U are incomparable with (resp. less than) all the elements of V. As it can be verified that U and V are good parts of C, the induction hypothesis applied on U, V, e_1 and e_2 permits to conclude. The other cases (*, $^{\omega}$, $^{-\omega}$, $\natural$, $^{-\natural}$ and $\diamond$) use extensions of this principle. Let us focus on the case $e = e'^{\omega}$ for example, the other induction steps being similar. Let D be a maximal block of C. Then $D \in L(e)$ iff there exist $J \in \mathcal{S}$ and a factorization of D into a sum of posets $D = \sum_{j \in J} D_j$, such that J is isomorphic to ω and $D_j \in L(e')$ for each

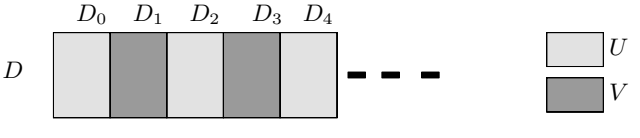


Fig. 2. A poset $D \in SP^{\diamond}(A)$ belongs to $L(e'^{\omega})$ iff it can be decomposed into $D = \sum_{j \in \omega} D_j$ where $D_j \in L(e')$ for all $j \in \omega$. Following this decomposition, $D \in L(e'^{\omega})$ iff it can be decomposed into $D = U \oplus V$, where each maximal block of U or V is a D_j for some $j \in \omega$. A linear ordering W isomorphic to ω can be built by taking one element in each maximal block of U and V.

$j \in J$. The linear ordering J can be partitioned itself into J_1 and J_2 such that if $x \in J_i$ then $x+1 \in J_k$ for any $x \in J$, where $i,k \in \{1,2\}$ and $i \neq k$. Thus, D can be partitioned into U and V such that $x \in U$ (resp. $x \in V$) iff $x \in D_j$ for some $j \in J_1$ (resp. $j \in J_2$). Finally, $D \in L(e)$ iff there exist U and V such that $D = U \oplus V$, each maximal (with respect to inclusion) block of U and V is a D_j for some $j \in J$, and J is isomorphic to ω. As it can be verified that each maximal block of U or V is a good part of P, the induction hypothesis can be applied. Figure 2 illustrates such a factorization of D.
The formula $\varphi_{e'^{\omega}}(X)$ is given below:

$$\begin{aligned}\varphi_{e'^{\omega}}(X) \equiv{}& \forall Z\ \mathtt{MaxBlock}(Z,X) \rightarrow \overline{\exists} U \overline{\exists} V\ Z = U \oplus V \wedge \varphi_{e_1}(U)\\ &\wedge \varphi_{e_1}(V) \wedge (\forall W\ \mathtt{Ordertype}(W,U,V) \rightarrow \mathtt{Omega}(W))\\ \mathtt{Ordertype}(Z,X,Y) \equiv{}& \forall B\ \mathtt{MaxBlock}(B,X) \rightarrow \exists! z\ z \in Z \cap B\\ &\wedge \forall B\ \mathtt{MaxBlock}(B,Y) \rightarrow \exists! z\ z \in Z \cap B\end{aligned}$$

For the other iteration operators, it suffices to change the test on W at the end of the formula $\varphi_{e'^{\omega}}$. For example, if the iteration operator is $^{\natural}$, $\mathtt{Omega}(W)$ is changed in order to check that W does not contain $-\omega$ as sub-ordering. The $\diamond$ composition of languages is processed using very similar arguments, and relies on Lemma 2, which gives another definition of $\diamond$, more adapted to a translation in the formalism of MSO[$<$] than the one of Section 3.

Lemma 2. *Let A be an alphabet and L_1 and L_2 be two languages of $SP^{\diamond}(A)$. Then $P \in L_1 \diamond L_2$ if and only if there exist a scattered and countable linear ordering $K \neq \emptyset$, a sequence $(P_k)_{k \in K}$ of posets of $SP^{\diamond}(A)$ and a map $f : K \rightarrow \{1,2\}$ such that the following conditions are true:*

1. *$P = \sum_{k \in K} P_k$;*
2. *if $f(k) = i$ and $k+1 \in K$ then $f(k+1) \neq i$;*
3. *if $k \in K$, k is not the last element of K, and k has no successor, then $f(k) = 2$;*
4. *if $k \in K$, k is not the first element of K, and k has no predecessor, then $f(k) = 2$;*
5. *if k is the first or the last element of K, then $f(k) = 1$;*
6. *K is complete;*
7. *$f(k) = i$ implies $P_k \in L_i$.*

In order to conclude, it suffices to observe that P is a good part and a maximal block of itself. Let $\psi_e \equiv \exists X(\forall x\ x \in X) \wedge \varphi_e(X)$. Then $L(e) = L(\psi_e)$. As a consequence:

Proposition 5. *Let A be an alphabet and $L \subseteq SP^{\diamond}(A)$. If L is series-rational, then $L = L(\psi_e)$ for any series-rational expression e such that $L = L(e)$.*

We finish by providing a last example.

Example 6. Let $A = \{a,b\}$ and $L \subseteq SP^{\diamond}(A)$ be the language of posets P of width at most 2, such that, if a a appears in a parallel part $P_1 \parallel P_2$ of P, then

$P_1 \parallel P_2$ is immediately followed by P_3 such that $wd(P_3) = 1$ and P_3 contains a b. Let $e_1 = (A + (b^\diamond \parallel b^\diamond))^\diamond + \epsilon$. The language of e_1 is all the posets that do not contain an a in parallel with another letter. Then L is the language of the series-rational expression

$$e = (e_1(((A^\diamond + \epsilon)a(A^\diamond + \epsilon)) \parallel A^\diamond)(A^\diamond + \epsilon)b)^\diamond e_1 + e_1$$

and of the second-order monadic sentence interpreted over $SP^\diamond(A)$

$$\phi_e \equiv \phi_{\leq 2} \wedge \forall x \forall y (R_a(x) \wedge x \parallel y) \rightarrow$$
$$\exists z\ x < z \wedge R_b(z) \wedge \mathtt{Flat}(z) \wedge \forall w\ x < w < z \rightarrow \mathtt{Flat}(w) \vee w \parallel y$$

where $\phi_{\leq 2} \equiv \forall X \mathtt{Antichain}(X) \rightarrow \vee_{i \leq 2} \mathtt{Card}_i(X)$ and $\mathtt{Flat}(x) \equiv \forall y\ x = y \vee x < y \vee y < x$.

We now turn to the definition of a finite $\parallel -\diamond$-semigroup recognizing L. Even when a $\parallel -\diamond$-semigroup S has a finite number of elements, its description is infinite because the result of the sequential product of any sequence of element of S must be given, and there are an infinity of them. A consequence is that finite $\parallel -\diamond$-semigroups do not really fit in the general framework of universal algebra. However, when the cardinal of S is finite, the sequential product can equivalently be replaced by three operations of finite arity: the sequential product of two elements, the ω repetition of an element and the reverse ω repetition. Formally, let $x \in S$:

$$x^\omega = \prod (s_i)_{i \in \omega} \text{ where } s_i = x \text{ for all } i \in \omega$$
$$x^{-\omega} = \prod (s_i)_{i \in -\omega} \text{ where } s_i = x \text{ for all } i \in -\omega$$

This technique was introduced by Wilke [24] in the case of finite ω-semigroups, and applied to the case of $\diamond$-semigroups in [8]. We use it in this example.

Let $S = \{a, b, ab, p, q, r, 0, 1\}$ be the finite $\parallel -\diamond$-semigroup defined by $a \parallel a = a \parallel b = a \parallel ab = ab \parallel b = ab \parallel ab = p$, $b \parallel b = (a \parallel b)b = q$, $b(b \parallel b) = b(a \parallel b)b = r$, $b(a \parallel b) = s$, $p \parallel x = q \parallel x = r \parallel x = 0$ for all $x \in S$ and by the binary sequential product, ω and $-\omega$ operations of Figure 3. It can easily be checked that S is depth-nilpotent. Let $\varphi : SP^\diamond(A) \rightarrow S$ be the morphism defined by $\varphi(a) = a$ and $\varphi(b) = b$. Then $L = \varphi^{-1}(\{a, b, ab, q, r, 1\})$.

$\cdot$	a	b	ab	p	q	r	s
a	a	ab	ab	p	q	r	s
b	ab	b	ab	s	r	r	s
ab	ab	ab	ab	s	r	r	s
p	p	q	q	0	0	q	p
q	q	q	q	p	q	q	p
r	r	r	r	s	r	r	s
s	s	r	r	0	0	r	s

	a	b	ab	p	q	r	s
ω	a	b	ab	0	q	r	r

	a	b	ab	p	q	r	s
$-\omega$	a	b	ab	0	q	q	p

Fig. 3. A finite description of the sequential product of S

7 Conclusion

The proof of the main result of the paper is effective. A consequence is that MSO[<] interpreted over $SP^{\diamond}_{\leq n}$ is decidable. As a corollary, any question on series-rational languages expressible in MSO[<], as for example the inclusion problem of series-rational languages, is also decidable.

Linear-rational expressions were introduced in [5] as well as equivalent automata. It is known from [2] that a language of scattered and countable linear structures is linear-rational if and only if it is in MSO[<]. When non-scattered linear structures are taken into account, a shuffle operation [4] must be added to linear-rational expressions to keep the equivalence with automata. Linear-rational languages with shuffle are a strict subset of the languages of MSO[<] interpreted over all countable linear structures.

A notion of automata accepting labelled posets of $SP^{\diamond}(A)$ have been introduced in [3], as well as equivalent rational expressions, called *series-parallel* rational expressions. The series-parallel rational expressions are an extension of the series-rational expressions, and automata equivalent to the latter, called *fork-acyclic*, are a restriction of automata of the former. The link between series-parallel rationality, series-rationality, automata and logic was first studied by Lodaya and Weil [14] and Kuske [11] for finite and ω structures. In this case, second-order definability coincides with series-rationality. One more assertion could be added to Theorem 3:

Theorem 4. *Let A be an alphabet and $L \subseteq SP^{\diamond}(A)$. Then the following assertions are equivalent:*

1. *L is series-rational,*
2. *there exists a morphism $\varphi : SP^{\diamond}(A) \to S$ into a finite depth-nilpotent $\parallel -\diamond$-semigroup S and $X \subseteq S$ such that $0 \notin X$, $L = \varphi^{-1}(X)$, and $\varphi(a) \neq 1$ for all $a \in A$,*
3. *L has bounded-width and is definable by a sentence of MSO[<],*
4. *L is accepted by a fork-acyclic automaton.*

Finite $\parallel -\diamond$-semigroups have important properties. One of them was used in Example 6: the sequential product can be equivalently replaced by operations whose descriptions are finite. This makes finite $\parallel -\diamond$-semigroups really finite algebraic objects. As another property, a particular finite $\parallel -\diamond$-semigroup can be canonically attached to any recognizable language L of posets of $SP^{\diamond}(A)$. This finite $\parallel -\diamond$-semigroup, called the *syntactic* $\parallel -\diamond$-semigroup of L, divides any $\parallel -\diamond$-semigroup recognizing L. In formal languages theory, this algebra plays an important role. The importance of syntactic algebras is emphasized for infinite structures: for example, contrary to the finite words case, a minimal automaton can not any more be attached to a recognizable language of infinite words. The existence of the syntactic $\parallel -\diamond$-semigroup of a recognizable language of $SP^{\diamond}(A)$ lays the foundation for the generalization of numerous results of formal languages of infinite words.

Finally, let us mention the study of Ésik and Németh [10] on rational sets of finite series-parallel posets, but with a non-commutative parallel composition.

In this case, as in the commutative parallel case, rationality, regularity, recognizability and logical definability all coincide for bounded-width languages.

Acknowledgements

The author wishes to express his thanks to the anonymous referees for helpful remarks that have been used to improve the quality of this paper.

References

1. Almeida, J.: Finite semigroups and universal algebra. Series in algebra, vol. 3. World Scientific, Singapore (1994)
2. Bedon, N., Bès, A., Carton, O., Rispal, C.: Logic and rational languages of words indexed by linear orderings. In: Hirsch, E.A., Razborov, A.A., Semenov, A., Slissenko, A. (eds.) Computer Science – Theory and Applications. LNCS, vol. 5010, pp. 76–85. Springer, Heidelberg (2008)
3. Bedon, N., Rispal, C.: Series-parallel languages on scattered and countable posets. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 477–488. Springer, Heidelberg (2007)
4. Bès, A., Carton, O.: A Kleene theorem for languages of words indexed by linear orderings. Int. J. Found. Comput. Sci. 17(3), 519–542 (2006)
5. Bruyère, V., Carton, O.: Automata on linear orderings. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 103–115. Springer, Heidelberg (2003)
6. Richard Büchi, J.: On a decision method in the restricted second-order arithmetic. In: Proc. Intern. Congr. on Logic, Methodology and Philosophy of Science, Berkeley 1960, pp. 1–11. Stanford University Press, Stanford (1962)
7. Büchi, J.R.: Transfinite automata recursions and weak second order theory of ordinals. In: Proc. Int. Congress Logic, Methodology, and Philosophy of Science, Jerusalem, pp. 2–23. North Holland, Amsterdam (1964)
8. Carton, O., Rispal, C.: Complementation of rational sets on countable scattered linear orderings. Int. J. Found. Comput. Sci. 16(4), 767–786 (2005)
9. Ebbinghaus, H.-D., Flum, J.: Finite model theory, 2nd edn. Springer monographs in mathematics. Springer, Heidelberg (1999)
10. Ésik, Z., Németh, Z.L.: Automata on series-parallel biposets. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 217–227. Springer, Heidelberg (2002)
11. Kuske, D.: Infinite series-parallel posets: logic and languages. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 648–662. Springer, Heidelberg (2000)
12. Kuske, D.: Towards a language theory for infinite N-free pomsets. Theoret. Comput. Sci. 299, 347–386 (2003)
13. Laüchli, H., Leonard, J.: On the elementary theory of linear order. Fund. Math. 59, 109–116 (1966)
14. Lodaya, K., Weil, P.: Series-parallel posets: algebra, automata and languages. In: Morvan, M., Meinel, C., Krob, D. (eds.) STACS 1998. LNCS, vol. 1373, pp. 555–565. Springer, Heidelberg (1998)
15. Lodaya, K., Weil, P.: Series-parallel languages and the bounded-width property. Theoret. Comput. Sci. 237(1–2), 347–380 (2000)

16. Lodaya, K., Weil, P.: Rationality in algebras with a series operation. Inform. Comput., 269–293 (2001)
17. Michaux, C., Point, F.: Les ensembles k-reconnaissables sont définissables dans $< N, +, V_k >$ (the k-recognizable sets are definable in $< N, +, V_k >$). C. R. Acad. Sci. Paris, Sér I(303), 939–942 (1986)
18. Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141, 1–35 (1969)
19. Rosenstein, J.G.: Linear Orderings. Academic Press, London (1982)
20. Shelah, S.: The monadic theory of order. Annals of Mathematics 102(3), 379–419 (1975)
21. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. III, pp. 389–455. Springer, Heidelberg (1997)
22. Valdes, J.: Parsing flowcharts and series-parallel graphs. Technical Report STAN-CS-78-682, Computer science departement of the Stanford University, Standford, Ca (1978)
23. Valdes, J., Tarjan, R.E., Lawler, E.L.: The recognition of series parallel digraphs. SIAM J. Comput. 11, 298–313 (1982)
24. Wilke, T.: An Eilenberg theorem for ∞-languages. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 588–599. Springer, Heidelberg (1991)

The Logic of Proofs as a Foundation for Certifying Mobile Computation

Eduardo Bonelli[1,2,⋆] and Federico Feller[2]

[1] Depto. de Ciencia y Tecnología, Universidad Nacional de Quilmes and CONICET
[2] LIFIA, Facultad de Informática, Universidad Nacional de La Plata

Abstract. We explore an intuitionistic fragment of Artëmov's *Logic of Proofs* as a type system for a programming language for *mobile units*. Such units consist of both a code and certificate component. Dubbed the *Certifying Mobile Calculus*, our language caters for both code and certificate development in a unified theory. In the same way that mobile code is constructed out of code components and extant type systems track local resource usage to ensure the mobile nature of these components, our system *additionally* ensures correct *certificate construction* out of certificate components. We present proofs of type safety and strong normalization for a run-time system based on an abstract machine.

1 Introduction

We explore an intuitionistic fragment (ILP) of Artëmov's *Logic of Proofs* (LP) as a type system for a programming language for *mobile units*. This language caters for both code and certificate development in a unified theory. LP may be regarded as refinement of modal logic S4 in which $\Box A$ is replaced by $[s]A$, for s a *proof term* expression, and is read: *"s is a proof of A"*. It is sound and complete w.r.t. provability in PA (see [Art95, Art01] for a precise statement) and realizes all theorems of S4. It therefore provides an answer to the (long-standing) problem of associating an exact provability semantics to S4 [Art95, Art01]. LP is purported to have important applications not only in logic but also in Computer Science [AB04]. This work may be regarded as a small step in exploring the applications of LP in programming languages and type theory.

Modal necessity $\Box A$ may be read as the type of programs that compute values of type A and that do not depend on local resources [Moo04, VCHP04, VCH05] or resources not available at the current stage of computation [TS97, WLPD98, DP01b]. The former reading refers to *mobile computation* ($\Box A$ as the type of mobile code that computes values of type A) while the latter to *staged computation* ($\Box A$ as the type of code that generates, at run-time, a program for computing a value of type A). See Sec. 7 for further references. We introduce the *Certifying Mobile Calculus* or $\lambda^{\mathsf{Cert}}_{\Box}$ by taking a mobile computation interpretation of ILP. ILP's mechanism for internalizing its own derivations provides a natural setting for code certification. A contribution of our approach

⋆ Work partially supported by Instituto Tecnológico de Buenos Aires.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 76–91, 2009.

is that, in the same way that mobile code is constructed out of code components and extant type systems track local resource usage to ensure the mobile nature of these components, our system *additionally* ensures correct *certificate construction* out of certificate components. Mobile units consist of both a code component and a certificate component. A sample $\lambda_{\Box}^{\mathsf{Cert}}$ expression, one encoding a proof of the ILP axiom scheme $[s](A \supset B) \supset [t]A \supset [s \cdot t]B$ where s, t are any proof term expressions and A, B any propositions, is the following:

$$\lambda a.\lambda b.unpack\ a\ to\ \langle u^{\bullet}, u^{\circ}\rangle\ in\ (unpack\ b\ to\ \langle v^{\bullet}, v^{\circ}\rangle\ in\ (box_{u^{\circ}\cdot v^{\circ}}\ u^{\bullet}v^{\bullet}))$$

This is read as follows: *"Given a mobile unit a and a mobile unit b, extract code $v^{\bullet}$ and certificate v° from b and extract code $u^{\bullet}$ and certificate u° from a. Then create new code $u^{\bullet}\,v^{\bullet}$ by applying $u^{\bullet}$ to $v^{\bullet}$ and a new certificate for this code $u^{\circ} \cdot v^{\circ}$. Finally, wrap both of these up into a new mobile unit."*. The syntax of code and certificates is described in detail in Sec. 3. The new mobile unit is created at the same current (implicit) world w. Moreover, the example assumes that both a and b reside at w. The following variant M illustrates the case where mobile units a and b reside at worlds w_a and w_b which are assumed different from the current world w:

$$unpack\, fetch[w_a]\, a\ to\ \langle u^{\bullet}, u^{\circ}\rangle\ in (unpack\ fetch[w_b]\, b\ to\ \langle v^{\bullet}, v^{\circ}\rangle\ in\ (box_{u^{\circ}\cdot v^{\circ}}\ u^{\bullet}v^{\bullet}))$$

Here the expression $fetch[w_a]\, a$ is operationally interpreted as a remote call to compute the value of a (a mobile unit) at w_a and then return it to the current world. Note that a and b occur free in this expression. Since b is a non-local resource it cannot be bound straightforwardly by prefixing the above term with λb. Rather, the code first must be moved from the current world w to w_b; similarly for a:

$$\lambda a.fetch[w_b]\,(\lambda b.fetch[w]\,M)$$

$\lambda_{\Box}^{\mathsf{Cert}}$ arises from a Curry-de Bruijn-Howard interpretation of a Natural Deduction presentation of ILP based on a judgemental analysis of the Logic of Proofs given in [AB07]. Propositions and proofs of ILP correspond to types and terms of $\lambda_{\Box}^{\mathsf{Cert}}$. Regarding semantics, we provide an operational reading of expressions encoding proofs in this system in terms of global computation. An abstract machine is introduced that computes over multiple worlds. Apart from the standard lambda calculus expressions new expressions for constructing mobile units and for computing in remote worlds are introduced. We state and prove *type safety* of a type system for $\lambda_{\Box}^{\mathsf{Cert}}$ w.r.t. its operational semantics. Also, we prove strong normalization.

This paper is organized as follows. Sec. 2 briefly recapitulates ILPnd [AB07], a Natural Deduction presentation of ILP. We then introduce a term assignment for ILPnd and discuss differences with the term assignment in [AB07] including the splitting of validity variables [AB07] into code and certificate variables. Sec. 4 introduces the run-time system of $\lambda_{\Box}^{\mathsf{Cert}}$, the abstract machine for execution of $\lambda_{\Box}^{\mathsf{Cert}}$ programs. Sec. 5 analyzes type safety and Sec. 6 strong normalization. References to related work follows. Finally, we conclude and suggest further directions for research. This is an extended abstract, full details may be found in a companion technical report [BF].

2 Natural Deduction for ILP

In previous work [AB07] a Natural Deduction presentation of ILP (ILPnd) is introduced by considering two sets of hypotheses, truth and validity hypotheses, and analyzing the meaning of the following Hypothetical Judgement with Explicit Evidence:

$$\Delta; \Gamma \rhd A \mid s$$

Here Δ is a sequence of *validity assumptions*, Γ a sequence of *truth assumptions*, A is a proposition and s is a proof term. A validity assumption is written $v : A$ where v ranges over a given infinite set of *validity variables* and states that A holds at all accessible worlds. Likewise, a truth assumption is written $a : A$ where a ranges over a given infinite set of *truth variables* and states that A holds at the current world. We write x to denote either of these variables. The judgement is read as: *"A is true with evidence s under validity assumptions Δ and truth assumptions Γ"*. Note that s is a constituent of this judgement without whose intended reading is not possible. The meaning of this judgement is given by axiom and inference schemes (Fig. 1). We say a judgement is *derivable* if it has a derivation using these schemes.

$$\begin{array}{lr} \textit{Proof Terms} & s,t ::= x \mid s \cdot t \mid \lambda a : A.s \mid !s \mid \text{LETC } s \text{ BE } v : A \text{ IN } t \\ \textit{Propositions} & A,B ::= P \mid A \supset B \mid [s]A \\ \textit{Truth Contexts} & \Gamma ::= \cdot \mid \Gamma, a : A \\ \textit{Validity Contexts} & \Delta ::= \cdot \mid \Delta, v : A \end{array}$$

Minimal Propositional Logic Fragment

$$\frac{}{\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a}\ \mathsf{oVar}$$

$$\frac{\Delta; \Gamma, a : A \rhd B \mid s}{\Delta; \Gamma \rhd A \supset B \mid \lambda a : A.s}\ \supset \mathsf{I} \qquad \frac{\Delta; \Gamma \rhd A \supset B \mid s \quad \Delta; \Gamma \rhd A \mid t}{\Delta; \Gamma \rhd B \mid s \cdot t}\ \supset \mathsf{E}$$

Provability Fragment

$$\frac{}{\Delta, v : A, \Delta'; \Gamma \rhd A \mid v}\ \mathsf{mVar}$$

$$\frac{\Delta; \cdot \rhd A \mid s}{\Delta; \Gamma \rhd [s]A \mid !s}\ \Box\mathsf{I} \qquad \frac{\Delta; \Gamma \rhd [r]A \mid s \quad \Delta, v : A; \Gamma \rhd C \mid t}{\Delta; \Gamma \rhd C\{v/r\} \mid \text{LETC } s \text{ BE } v : A \text{ IN } t}\ \Box\mathsf{E}$$

$$\frac{\Delta; \Gamma \rhd A \mid s \quad \Delta; \Gamma \vdash s \equiv t : A}{\Delta; \Gamma \rhd A \mid t}\ \mathsf{EqEvid}$$

Fig. 1. Explanation for Hypothetical Judgements with Explicit Evidence

All free occurrences of a (resp. v) in s are bound in $\lambda a : A.s$ (resp. $\textsc{letc}\, t\, \textsc{be}\, v : A\, \textsc{in}\, s$). A proposition is either a variable P, an implication $A \supset B$ or a validity proposition $[s]A$. We write "$\cdot$" for empty contexts and $s\{x/t\}$ for the result of substituting all free occurrences of x in s by t (bound variables are renamed whenever necessary); likewise for $A\{x/t\}$.

A brief informal explanation of some of these schemes follows. The axiom scheme oVar states that the judgement $\Delta; \Gamma, a : A, \Gamma' \rhd A \mid a$ is evident in itself. Indeed, if we assume that a is evidence that proposition A is true, then we immediately conclude that A is true with evidence a. The introduction scheme for the $[s]$ modality internalizes meta-level evidence into the object-logic. It states that if s is unconditional evidence that A is true, then A is in fact valid with witness s (i.e. $[s]A$ is true). Evidence for the truth of $[s]A$ is constructed from the (verified) evidence that A is unconditionally true by prefixing it with a bang constructor. Finally, $\Box\mathsf{E}$ allows the discharging of validity hypotheses. In order to discharge the validity hypotheses $v : A$, a proof of the validity of A is required. In this system, this requires proving that $[r]A$ is true with evidence s, for some evidence of proof r and s. Note that r is evidence that A is unconditionally true (i.e. valid) whereas s is evidence that $[r]A$ is true. The former is then substituted in the place of all free occurrences of v in the proposition C. This construction is recorded with evidence $\textsc{letc}\, s\, \textsc{be}\, v : A\, \textsc{in}\, t$ in the conclusion.

Since ILPnd internalizes its own derivations and normalization introduces identities on derivations at the meta-level, such identities must be reflected in the object-logic too. This is the aim of EqEvid. The schemes defining the judgement of evidence equality $\Delta; \Gamma \vdash s \equiv t : A$ are the axioms for β equality and β equality on $\Box$ together with appropriate congruence schemes (consult [AB07] for details). It should be noted that soundness of ILPnd with respect to ILP does not require the presence of EqEvid. It is, however, required in order for normalization to be closed over the set of derivations.

A sample derivation in ILPnd of $[s](A \supset B) \supset [t]A \supset [s \cdot t]B$ follows, where $\Gamma = a : [s](A \supset B), b : [t]A$ and $\Delta = u : A \supset B, v : A$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot; \Gamma \rhd [s](A \supset B) \mid a} \quad \dfrac{\dfrac{\dfrac{}{\Delta; \cdot \rhd A \supset B \mid u} \quad \dfrac{}{\Delta; \cdot \rhd A \mid v}}{\Delta; \cdot \rhd B \mid u \cdot v}\,\Box\mathsf{I}}{\Delta; \Gamma \rhd [u \cdot v]B \mid !(u \cdot v)}}{u : (A \supset B); \Gamma \rhd [u \cdot t]B \mid \textsc{letc}\, b\, \textsc{be}\, v : A\, \textsc{in}\, !(u \cdot v)}\,\Box\mathsf{E}}{\cdot; \Gamma \rhd [s \cdot t]B \mid \textsc{letc}\, a\, \textsc{be}\, u : A \supset B\, \textsc{in}\, \textsc{letc}\, b\, \textsc{be}\, v : A\, \textsc{in}\, !u \cdot v}\,\Box\mathsf{E}}{\cdot; a : [s](A \supset B) \rhd [t]A \supset [s \cdot t]B \mid \lambda b : [s]A.\textsc{letc}\, a\, \textsc{be}\, u : A \supset B\, \textsc{in}\, \textsc{letc}\, b\, \textsc{be}\, v : A\, \textsc{in}\, !(u \cdot v)}\,\supset\mathsf{I}}{\cdot; \cdot \rhd [s](A \supset B) \supset [t]A \supset [s \cdot t]B \mid \lambda a : [s](A \supset B).\lambda b : [t]A.\textsc{letc}\, a\, \textsc{be}\, u : A \supset B\, \textsc{in}\, \textsc{letc}\, b\, \textsc{be}\, v : A\, \textsc{in}\, !(u \cdot v)}\,\supset\mathsf{I}$$

3 Term Assignment

We assume a set $\{w_1, w_2, \ldots\}$ of worlds, a set $\{v_1^\bullet, v_2^\bullet, \ldots\}$ of code variables and a set $\{v_1^\circ, v_2^\circ, \ldots\}$ of certificate variables. We use Σ for a (finite) set of worlds.

Δ and Γ are as before. The syntactic categories of *certificates*, *values* and *terms* are defined as follows:

$$\begin{aligned} s, t &::= a \mid v^\circ \mid s \cdot t \mid \lambda a : A.s \mid !s \mid \mathit{letc}\ s\ \mathit{be}\ v^\circ : A\ \mathit{in}\ t \mid \mathit{fetch}(s) \\ V &::= \mathit{box}_s\, M \mid \lambda a.M \\ M, N &::= a \mid v^\bullet \mid V \mid M\, N \\ &\quad \mid \ \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N \mid \mathit{fetch}[w]\, M \end{aligned}$$

Certificates have two kinds of variables. Local variables a are used for abstracting over local assumptions when constructing certificates. Certificate variables v° represent unknown certificates. $s \cdot t$ is certificate composition. $!s$ is certificate endorsement. $\mathit{letc}\ s\ \mathit{be}\ v^\circ : A\ \mathit{in}\ t$ is certificate validation, the inverse operation to endorsement. Finally, $\mathit{fetch}(s)$ certifies the *fetch* code movement operation to be described shortly. Substitution of code variables for terms in terms ($M\{v^\bullet/N\}$) and substitution of certificate variables for certificates in certificates ($t\{v^\circ/s\}$) and in terms ($M\{v^\circ/s\}$) is defined as expected. An example of a certificate is the following, which encodes a derivation of the first example presented in the introduction:

$$\lambda a : [s](A \supset B).\lambda b : [t]A.\mathit{letc}\ a\ \mathit{be}\ u^\circ : A \supset B\ \mathit{in}\ (\mathit{letc}\ b\ \mathit{be}\ v^\circ : A\ \mathit{in}\ !(u^\circ \cdot v^\circ))$$

Values are a subset of terms that represent the result of computations of well-typed, closed terms. A value of the form $\lambda a.M$ is an abstraction (free occurrences of a in M are bound as usual) and one of the form $\mathit{box}_s\, M$ is a *mobile unit* (composed of mobile code M and certificate s). A *term* is either a term variable for local code a, a term variable for mobile code $v^\bullet$, a value V, an application term $M\, N$, an unpacking term for extraction of code-certificate pairs from mobile units $\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ \rangle\ \mathit{in}\ N$ (free occurrences of v° and $v^\bullet$ in N are bound by this construct) or a fetch term $\mathit{fetch}[w]\, M$. In an unpacking term, M is the argument and N is the body; in a fetch term we refer to w as the target of the *fetch* and M as its body. The operational semantics of these constructs is discussed in Sec. 4.

The term assignment results essentially (the differences are explained below) from the schemes of Fig. 1 with terms encoding derivations and localizing the hypotheses in Δ, Γ at specific worlds. Also, a reference to the current world is added. Typing judgements take the form

$$\Sigma; \Delta; \Gamma \rhd M : A@w \mid s \tag{1}$$

Validity and truth contexts are now sequences of expressions of the form $v : A@w$ and $a : A@w$, respectively. The former indicates that mobile unit v computing a value of type A may be assumed to exist and to be located at world w. The latter indicates that a local value a of type A may be assumed to exist at world w. The truth of a proposition at w shall rely, on the one hand, on truth hypotheses in Γ that are located at w, and on the other, on validity hypotheses in Δ that have been fetched, from their appropriate hosts, to the current location w. Logical connectives bind tighter than @, therefore an expression such as $A \supset B@w$ should be read as $(A \supset B)@w$.

It should be mentioned that ILP is not a hybrid logic [AtC06]. In other words, $A@w$ is not a proposition of our object-logic. For example, expressions of the form $A@w \supset B@w'$ are not valid propositions.

3.1 Typing Schemes

Typing schemes defining (1) are presented in Fig. 2 and discussed below. A first difference with ILPnd is that the scheme EqEvid has been dropped. Although the latter is required for normalization of derivations to be a closed operation (as already mentioned), our operational interpretation of terms does not rely on normalization of Natural Deduction proofs. For a computational interpretation of ILP based on normalization the reader may consult [AB07]. A further difference is that $\Box$I has been refined into two schemes, namely $\Box I$ and Fetch. The first introduces a modal formula and states it to be true at the current world w. The second states that all worlds accessible to w may also assume this formula to be true.

In this work mobile code is accompanied by a certificate. We speak of *mobile units* rather than mobile code to emphasize this. Since mobile units are expressions of modal types and validity variables v represent holes for values of modal types, validity variables v may actually be seen as pairs $\langle v^\bullet, v^\circ \rangle$. Here $v^\bullet$ is the mobile code component and v° is the certificate component of the mobile unit[1]. As a consequence, the modality axiom mVar of ILPnd now takes the following form, where judgement $\Sigma \vdash w$ ensures w is a world in Σ (it is defined by requiring $w \in \Sigma$):

$$\frac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^\bullet : A@w \mid v^\circ}\ \mathrm{VarV}$$

The schemes $\supset I$ and $\supset E$ form abstractions and applications at the current world w. Applications of these schemes are reflected in their corresponding certificates. Scheme $\Box I$ states that if we have a typing derivation of M that does not depend on local assumptions (although it may depend on assumptions universally true) and s is a witness to this fact, then M is in fact executable at an arbitrary location. Thus a mobile unit $box_s\, M$ is introduced. The Fetch scheme types the *fetch* instruction. A term of the form $fetch[w']\, M$ at world w is typed by considering M at world w'. We are in fact assuming that w sees w' (or that w' is accessible from w) at run-time. Moreover, since the result of this instruction is to compute M at w' and then *return* the result to w (cf. Sec. 4), worlds w' and w are assumed interaccessible[2]. The *unpack* instruction is typed using the scheme $\Box E$. Suppose we are given a term N that computes some value of type C at world w and depends on a validity hypotheses $v : A@w$. Suppose we also

[1] The "$\circ$" is reminiscent of a wrapping with which the interior "$\bullet$" is protected. Hence our use of the former for certificates and the latter for code.

[2] We are considering a term assignment for a Natural Deduction presentation of a refinement of S4 (and not S5; see Lem. 3). This reading, which suggests symmetry of the accessibility relation in a Kripke style model (and hence S5), is part of the run-time interpretation of terms (cf. 7).

$$\frac{\Sigma \vdash w}{\Sigma; \Delta; \Gamma, a : A@w, \Gamma' \rhd a : A@w \,|\, a}\ \mathrm{VarT}$$

$$\frac{\Sigma; \Delta; \Gamma, a : A@w \rhd M : B@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd \lambda a.M : A \supset B@w \,|\, \lambda a : A.s} \supset I \qquad \frac{\Sigma; \Delta; \Gamma \rhd M : A \supset B@w \,|\, s \quad \Sigma; \Delta; \Gamma \rhd N : A@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd M\, N : B@w \,|\, s \cdot t} \supset E$$

$$\frac{\Sigma \vdash w}{\Sigma; \Delta, v : A@w, \Delta'; \Gamma \rhd v^\bullet : A@w \,|\, v^\circ}\ \mathrm{VarV}$$

$$\frac{\Sigma; \Delta; \cdot \rhd M : A@w \,|\, s}{\Sigma; \Delta; \Gamma \rhd box_s\, M : [s]A@w \,|\, !s} \Box I \qquad \frac{\Sigma; \Delta; \Gamma \rhd M : [s]A@w' \,|\, t \quad \Sigma \vdash w}{\Sigma; \Delta; \Gamma \rhd fetch[w']\, M : [s]A@w \,|\, fetch(t)}\ \mathrm{Fetch}$$

$$\frac{\Sigma; \Delta; \Gamma \rhd M : [r]A@w \,|\, s \quad \Sigma; \Delta, v : A@w; \Gamma \rhd N : C@w \,|\, t}{\Sigma; \Delta; \Gamma \rhd unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N : C\{v^\circ/r\}@w \,|\, letc\ s\ be\ v : A\ in\ t} \Box E$$

Fig. 2. Term assignment for ILPnd

have a term M that computes a mobile unit of type $[r]A@w$ at the same world w. Then *unpack* M *to* $\langle v^\bullet, v^\circ \rangle$ *in* N is well-typed at w and computes a value of type $C\{v^\circ/r\}$. The certificate *letc* s *be* $v : A$ *in* t encodes the application of this scheme.

The following substitution principles reveal the true hypothetical nature of hypotheses, both for truth and for validity. Both are proved by induction on the derivation of the second judgement.

Lemma 1 (Substitution principle for truth hypotheses). *If* $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd M : A@w \,|\, s$ *and* $\Sigma; \Delta; \Gamma_1, a : A@w, \Gamma_2 \rhd N : B@w' \,|\, t$ *are derivable, then so is* $\Sigma; \Delta; \Gamma_1, \Gamma_2 \rhd N\{a/M\} : B@w' \,|\, t\{a/s\}$.

Lemma 2 (Substitution principle for validity hypotheses). *If* $\Sigma; \Delta_1, \Delta_2; \cdot \rhd M : A@w \,|\, s$ *and* $\Sigma; \Delta_1, v : A@w, \Delta_2; \Gamma \rhd N : B@w' \,|\, t$ *are derivable, then so is* $\Sigma; \Delta_1, \Delta_2; \Gamma \rhd N\{v^\circ/s\}\{v^\bullet/M\} : B\{v^\circ/s\}@w \,|\, t\{v^\circ/s\}$.

Regarding the relation of this type system for $\lambda^{\mathsf{Cert}}_{\Box}$ with ILPnd we have the following result, which may be verified by structural induction on the derivation of the first judgement. Applications of the Fetch scheme become instances of the scheme $\frac{\mathcal{J}}{\mathcal{J}}$ with copies of identical judgements in ILPnd.

Lemma 3. *If* $\Sigma; \Delta; \Gamma \rhd A@w \,|\, s$ *is derivable, then so is* $\Delta'; \Gamma' \rhd A' \mid s'$ *in ILPnd, where* Δ' *and* Γ' *result from* Δ *and* Γ*, respectively, by dropping all location qualifiers and* A' *and* s' *result from* A *and* s*, respectively, by replacing all occurrences of* $v^\bullet$ *and* v° *by* v *and replacing all certificates of the form* $fetch(s)$ *with* s.

4 Operational Semantics

The operational semantics of $\lambda^{\mathsf{Cert}}_{\Box}$ follows ideas from [VCHP04]. We introduce an abstract machine over a network of nodes. Nodes are named using worlds.

Computation takes place sequentially, at some designated world. We are, in effect, modelling sequential programs that are aware of other worlds (other than their local host), rather than concurrent computation. An *abstract machine state* is an expression of the form $\mathbb{W}; w : [k, M]$ (top of Fig. 3). The world w indicates the node where computation is currently taking place. M is the code that is being executed under local context k (M is the current *focus* of computation). The context k is a stack of terms with holes (written "$\circ$") that represent the layers of terms that are peeled out in order to access the redex. This representation ensures a reduction relation that always operates at the root of an expression and thus allows us to speak of an abstract machine. An alternative presentation based on a small or big-step semantics on terms, rather than machine states, is also possible. Continuing our explanation of the context k, it is a sequence of terms with holes ending in either return w or finish. return w indicates that once the term currently in focus is computed to a value, this value is to be returned to world w. The type system ensures that this value is, in effect, a mobile unit. If k takes the form finish, then the value of the term currently in focus is the end result of the computation. Finally, $k \triangleleft l$ states that the outermost peeled term layer is l. This latter expression may be of one of the following forms: $\circ\ N$ indicates a pending argument, $V\ \circ$ a pending abstraction (that V is an abstraction rather than a mobile unit is enforced by the type system) and $unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N$ a pending unpack body.

Finally, $\mathbb{W}$ is called a *network environment* and encodes the current state of execution at the remaining nodes of the network. The *domain of* $\mathbb{W}$ is the set of worlds to which it refers. Also, we sometimes refer to $\mathbb{W}; k$ as the network environment.

The *initial* machine state (over $\Sigma = \{w_1, \ldots, w_n\}$) is $\mathbb{W}; w : [\text{finish}, M]$, where $\mathbb{W} = \{w_1 : \epsilon, \ldots, w_n : \epsilon\}$ and w and M are any world and term, respectively. Similarly, the *terminal* machine state is one of the form $\mathbb{W}; w : [\text{finish}, V]$. Note that in a terminal state the focus of computation is a fully evaluated term (i.e. a value).

Run – time system syntax

$$
\begin{array}{rl}
\mathbb{N} & ::= \mathbb{W}; w : [k, M] \\
\mathbb{W} & ::= \{w_1 : C_1, \ldots w_n : C_n\} \\
k & ::= \text{return } w \mid \text{finish} \mid k \triangleleft l \\
l & ::= \circ\ N \mid V\ \circ \mid unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N \\
C & ::= \epsilon \mid C :: k
\end{array}
$$

Run – time system reduction schemes

$$
\begin{array}{lr}
(1) & \mathbb{W}; w : [k, MN] \longrightarrow \mathbb{W}; w : [k \triangleleft \circ\ N, M] \\
(2) & \mathbb{W}; w : [k \triangleleft \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \triangleleft V\ \circ, N] \\
(3) & \mathbb{W}; w : [k \triangleleft (\lambda a.M)\ \circ, V] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}] \\
(4) & \mathbb{W}; w : [k, unpack\ M\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N] \longrightarrow \mathbb{W}; w : [k \triangleleft unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, M] \\
(5) & \mathbb{W}; w : [k \triangleleft unpack\ \circ\ to\ \langle v^\bullet, v^\circ \rangle\ in\ N, box_s\ M] \longrightarrow \mathbb{W}; w : [k, N\{v^\circ/s\}\{v^\bullet/M\}] \\
(6) & \{w : C; w_s\}; w : [k, fetch[w']\ M] \longrightarrow \{w : C :: k; w_s\}; w' : [\text{return } w, M] \\
(7) & \{w : C :: k; w_s\}; w' : [\text{return } w, V] \longrightarrow \{w : C; w_s\}; w : [k, V]
\end{array}
$$

Fig. 3. Operational semantics of $\lambda_{\square}^{\mathsf{Cert}}$

4.1 Reduction Schemes

The operational semantics is presented by means of a small-step call-by-value reduction relation whose definition is given by the *reduction schemes* depicted in Fig. 3. The first scheme selects the leftmost term in an application for reduction and pushes the pending part of the term (in this case the argument of the application) into the context. Once a value is attained (which the type system, described below, will ensure to be an abstraction) the pending argument is popped off the context for reduction and the value V is pushed onto the context. Finally, when the argument has been reduced to a value, the pending abstraction is popped off the context and the beta reduct placed into focus for the next computation step. In the case that reduction encounters an *unpack* term, the argument M is placed into focus whilst the rest of the the term is pushed onto the context. When reduction of the argument of an *unpack* computes a value, more precisely a mobile unit, the code and certificate components are extracted from it and substituted in the body of the *unpack* term. Note that the schemes presented up to this point all compute locally, we now address those that operate non-locally. If a computation's focus is on a *fetch* instruction, then the execution context k is pushed onto the network environment for the current world w' and control transfers to world w. Moreover, focus of computation is now placed on the term M. Finally, the context of computation at w is set to return w thus ensuring that, once a value is computed, control transfers back to the caller. The latter is the rôle of the final reduction scheme.

5 Type Soundness

This section addresses both *progress* (well-typed, non-terminal machine states are not stuck) and *subject reduction* (well-typed machine states are closed under the reduction). Recall from above that a machine state $\mathbb{N}$ is *terminal* if it is of the form $\mathbb{W}; w : [\text{finish}, V]$. It is *stuck* if it is not terminal and there is no $\mathbb{N}'$ such that $\mathbb{N} \longrightarrow \mathbb{N}'$. Two new judgements are introduced, machine state judgements and network environment judgements:

- $\Sigma \vdash \mathbb{W}; w_j : [k, M]$
- $\Sigma \vdash \mathbb{W}; k : A@w_j$

The first states that $\mathbb{W}; w_j : [k, M]$ is a well-typed machine state under the set of worlds Σ. The second states that the network environment together with the local context is well-typed under the set of worlds Σ.

A machine state is well-typed (Fig. 4) if the following three requirements hold. First $\mathbb{W}$ is a network environment with domain Σ. Second, M is closed, well-typed code at world w_j with certificate s that produces a value of type A, if at all. Finally, the network environment should be well-typed. The type of $\mathbb{W}$; finish has to be the type of the term currently in focus and located at the same world as indicated in the machine state. A network environment $\mathbb{W}; k \triangleleft \circ\, N$ is well-typed with type $A \supset B$ at world w under Σ, if the argument is well-typed with type A at w, and

$$\frac{}{\Sigma \vdash \mathbb{W}\,;\mathrm{finish} : A@w}\ \mathsf{C.Finish}$$

$$\frac{\Sigma \vdash \mathbb{W}\,;k : B@w \quad \Sigma;\cdot\,;\cdot \rhd N : A@w \mid s}{\Sigma \vdash \mathbb{W}\,;k \lhd \circ\ N : A \supset B@w}\ \mathsf{C.Abs} \qquad \frac{\Sigma \vdash \mathbb{W}\,;k : B@w \quad \Sigma;\cdot\,;\cdot \rhd V : A \supset B@w \mid s}{\Sigma \vdash \mathbb{W}\,;k \lhd V\ \circ : A@w}\ \mathsf{C.App}$$

$$\frac{\Sigma \vdash \mathbb{W}\,;k : B\{v^\circ/t\}@w \quad \Sigma;v : A;\cdot \rhd N : B@w \mid s}{\Sigma \vdash \mathbb{W}\,;k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N : [t]A@w}\ \mathsf{C.Box}$$

$$\frac{\Sigma \vdash \{w' : C; w_s\}\,;k : A@w'}{\Sigma \vdash \{w' : C :: k; w_s\}\,;\mathrm{return}\ w' : A@w}\ \mathsf{C.Return}$$

$$\frac{\Sigma = \{w_1,\ldots,w_n\} \quad \mathbb{W} = \{w_1 : C_1, \ldots w_n : C_n\} \quad \Sigma;\cdot\,;\cdot \rhd M : A@w_j \mid s \qquad \Sigma \vdash \mathbb{W}\,;k : A@w_j}{\Sigma \vdash \mathbb{W}\,;w_j : [k, M]}\ \mathsf{MState}$$

Fig. 4. Typing schemes for machine states

the network environment $\mathbb{W};k$ is well-typed with type B at the same world and under the same set of worlds. Note that $A \supset B$ is the type of the hole in the next term layer in k, and shall be completed by applying the term in focus to N. This is reminiscent of the left introduction scheme for implication in the Sequent Calculus presentation of Intuitionistic Propositional Logic. This connection is explored in detail in [Her94, CH00]. The C.App and C.Box schemes may be described in similar terms. Regarding the judgement $\Sigma \vdash \{w' : C :: k; w_s\}\,;\mathrm{return}\ w' : A@w$, in order to verify that the type A at w of the value to be returned to world w' is correct, the context at w' must be checked, at w', to see if its outermost hole is indeed expecting a value of this type.

We now state the promised results. Both are proved by structural induction on the derivation of the judgement $\Sigma \vdash \mathbb{N}$. Together these results imply soundness of the reduction relation w.r.t. the type system: if a machine state is typable under Σ and is not terminal, then a well-typed value shall be attained.

Proposition 1 (Progress). *If $\Sigma \vdash \mathbb{N}$ is derivable and $\mathbb{N}$ is not terminal, then there exists $\mathbb{N}'$ such that $\mathbb{N} \longrightarrow \mathbb{N}'$.*

Proposition 2 (Subject Reduction). *If $\Sigma \vdash \mathbb{N}$ is derivable and $\mathbb{N} \longrightarrow \mathbb{N}'$, then $\Sigma \vdash \mathbb{N}'$ is derivable.*

6 Strong Normalization

We prove strong normalization (SN) of machine reduction by translating machine states to terms of the simply typed lambda calculus with unit type ($\lambda^{\mathbf{1},\rightarrow}$). For technical reasons (which we comment on shortly) we shall consider the following modification of the machine reduction semantics of $\lambda^{\mathsf{Cert}}_{\Box}$ obtained by replacing the reduction scheme:

$$(2)\ \mathbb{W};w : [k \lhd \circ\ N, V] \longrightarrow \mathbb{W};w : [k \lhd V\ \circ, N]$$

$$\text{Machine reduction } (\lambda^{\mathsf{Cert}}_{\Box}) \xrightarrow[F(\cdot)]{} \text{Lambda reduction } (\lambda^{\mathsf{Cert}}_{\Box}) \xrightarrow[T(\cdot)]{} \text{Simply typed lambda calculus } (\lambda^{\mathbf{1},\to})$$

$$\begin{aligned}
F(\mathbb{W}; w : [\text{finish}, M]) &=_{def} M \\
F(\mathbb{W}; w : [k \lhd \circ\ N, M]) &=_{def} F(\mathbb{W}; w : [k, M\ N]) \\
F(\mathbb{W}; w : [k \lhd V\ \circ, N]) &=_{def} F(\mathbb{W}; w : [k, V\ N]) \\
F(\mathbb{W}; w : [k \lhd \mathit{unpack}\ \circ\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N, M]) &=_{def} F(\mathbb{W}; w : [k, \mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N]) \\
F(\{w : C :: k; w_s\}; w' : [\text{return } w, M]) &=_{def} F(\{w : C; w_s\}; w : [k, M])
\end{aligned}$$

$$\begin{aligned}
T(P) &=_{def} P & T(a) &=_{def} a \\
T(A \supset B) &=_{def} T(A) \supset T(B) & T(v^\bullet) &=_{def} v\ \mathit{unit} \\
T([s]A) &=_{def} \mathbf{1} \supset T(A) & T(\lambda a.M) &=_{def} \lambda a.\, T(M) \\
& & T(M\ N) &=_{def} T(M)\ T(N) \\
& & T(\mathit{box}_s\ M) &=_{def} \lambda a.\, T(M), a \text{ fresh of type } \mathbf{1} \\
& & T(\mathit{unpack}\ M\ \mathit{to}\ \langle v^\bullet, v^\circ\rangle\ \mathit{in}\ N) &=_{def} (\lambda v.\, T(N))\ T(M) \\
& & T(\mathit{fetch}[w]\ M) &=_{def} (\lambda a.a)\ T(M)
\end{aligned}$$

Fig. 5. From machine reduction to the simply typed lambda calculus

by the following two new reduction schemes:

$$\begin{aligned}
&(2.1) \qquad \mathbb{W}; w : [k \lhd \circ\ N, V] \longrightarrow \mathbb{W}; w : [k \lhd V\ \circ, N],\ N \text{ is not a value} \\
&(2.2)\ \mathbb{W}; w : [k \lhd \circ\ V, \lambda a.M] \longrightarrow \mathbb{W}; w : [k, M\{a/V\}]
\end{aligned}$$

These schemes result from refining (2) by inspecting its behavior in any non-terminating reduction sequence. If N happens to be a value, then each (2) step is followed by a (3) step. The juxtaposition of these two steps gives precisely (2.2). The reduction scheme (2.1) is just (2) when N is not a value. It is clear that every non-terminating reduction sequence in the original formulation can be mimicked by a non-terminating reduction sequence in the modified semantics in such a way that for each (2) step

- either it is not followed by a (3) step and thus becomes a (2.1) step or
- it is followed by a (3) step and hence (2) followed by (3) become one (2.2) step.

Therefore, it suffices to prove SN of the modified system in order to deduce the same property for our original formulation.

The proof of SN proceeds in two phases (Fig. 5). First we relate machine reduction with a notion of reduction that operates directly on lambda terms via a mapping $F(\cdot)$. Then we relate the latter with reduction in $\lambda^{\mathbf{1},\to}$ via a mapping $T(\cdot)$. We consider the first phase. The map $F(\cdot)$ flattens out the local context of a machine state in order to produce a term of $\lambda^{\mathsf{Cert}}_{\Box}$. This function is type preserving, a result which is proved by induction on the pair $\langle|\mathbb{W}|, k\rangle$, where $|\mathbb{W}|$ is the size of $\mathbb{W}$ (i.e. the sum of the length of the context stacks of all worlds in its domain).

Lemma 4. *Let $\mathbb{N}$ be $\mathbb{W}; w : [k, M]$. If $\Sigma \vdash \mathbb{N}$ is derivable, then there exist A and s such that $\Sigma; \cdot\,; \cdot \rhd F(\mathbb{N}) : A@w \,|\, s$ is derivable.*

In order to relate machine reduction in $\lambda_{\Box}^{\mathsf{Cert}}$ with reduction in $\lambda^{\mathbf{1},\rightarrow}$ we introduce *lambda reduction*. These schemes are standard except for the last one which states that *fetch* terms have no computational effect at the level of lambda terms. It should be mentioned that strong lambda reduction reduction is considered (i.e. reduction under all term constructors).

Definition 1 (Lambda reduction for $\lambda_{\Box}^{\mathsf{Cert}}$).

$$\begin{array}{rl} (\lambda a.M)\, N & \longrightarrow_{\beta} \quad M\{a/N\} \\ \mathit{unpack}\ \mathit{box}_s\, M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N & \longrightarrow_{\beta_{\Box}} \quad N\{v^{\bullet}/M\}\{v^{\circ}/s\} \\ \mathit{fetch}[w]\, M & \longrightarrow_{\mathit{ftch}} M \end{array}$$

We can now establish that the flattening map is also reduction preserving:

Lemma 5. *If* $\mathbb{N} \longrightarrow_{1,2.1,4,7} \mathbb{N}'$, *then* $F(\mathbb{N}) = F(\mathbb{N}')$.
If $\mathbb{N} \longrightarrow_{2.2,3,5,6} \mathbb{N}'$, *then* $F(\mathbb{N}) \longrightarrow_{\beta,\beta_{\Box},\mathit{ftch}} F(\mathbb{N}')$.

The second part of the proof consists in relating lambda reduction in $\lambda_{\Box}^{\mathsf{Cert}}$ with reduction in $\lambda^{\mathbf{1},\rightarrow}$. For that we introduce a mapping $T(\cdot)$ (Fig. 5) that associates types and terms in $\lambda_{\Box}^{\mathsf{Cert}}$ with types and terms in $\lambda^{\mathbf{1},\rightarrow}$. Function types are translated to function types and the modal type $[s]A$ is translated to functional types whose domain is the unit type $\mathbf{1}$ and whose codomain is the translation of A. Translation of terms is straightforward given the translation on types; the case for *fetch* guarantees that each $\longrightarrow_{\mathit{ftch}}$ step is mapped to a non-empty step in $\lambda^{\mathbf{1},\rightarrow}$. $T(\cdot)$ over terms is both type preserving and reduction preserving. The first of these is proved by induction over the derivation of $\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$.

Lemma 6. *If* $\Sigma; \Delta; \Gamma \rhd M : A@w \,|\, s$ *is derivable in* $\lambda_{\Box}^{\mathsf{Cert}}$, *then* $\Delta', \Gamma' \rhd T(M) : T(A)$ *is derivable in* $\lambda^{\mathbf{1},\rightarrow}$, *where*

1. Γ' *results from replacing each hypothesis* $a : A@w$ *by* $a : T(A)$ *and*
2. Δ' *results from replacing each hypothesis* $v : A@w$ *by* $v : \mathbf{1} \supset T(A)$.

The second is proved by induction on M making use of the fact that T commutes with substitution of (the translation of) local variables (i.e. $T(M)\{a/T(N)\} = T(M\{a/N\})$). T does not commute with substitution of (the translation of) validity variables (i.e. $T(M)\{v/T(N)\} \neq T(M\{v/N\})$; take $M = v^{\bullet}$). However, the following does hold and suffices for our purposes: $T(M)\{v/\lambda a.T(N)\} \longrightarrow_{\beta}^{*} T(M\{v^{\bullet}/N\}\{v^{\circ}/s\})$. The arrow $\longrightarrow_{\beta}^{*}$ denotes the reflexive, transitive closure of $\longrightarrow_{\beta}$ while $\longrightarrow_{\beta}^{+}$ (below) denotes its transitive closure.

Lemma 7. *If* $M \longrightarrow_{\beta,\beta_{\Box},\mathit{ftch}} N$, *then* $T(M) \longrightarrow_{\beta}^{+} T(N)$

Our desired result may be proved by contradiction as follows. Let us assume, for the time being, that $\longrightarrow_{1,2.1,4,7}$ reduction is SN. Suppose, also, that there is an infinite reduction sequence starting from a machine state $\mathbb{N}_1$. From our assumption this sequence must have an infinite number of interspersed $\longrightarrow_{2.2,3,5}$ reduction steps:

$$\mathbb{N}_1 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_2 \longrightarrow_{2.2,3,5} \mathbb{N}_3 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_4 \longrightarrow_{2.2,3,5} \mathbb{N}_5 \longrightarrow^*_{1,2.1,4,7} \mathbb{N}_6 \longrightarrow_{2.2,3,5} \dots$$

Then (Lem. 5) we have the following lambda reduction sequence over typable terms (Lem. 4):

$$F(\mathbb{N}_1) = F(\mathbb{N}_2) \longrightarrow_{\beta,\beta_\square,ftch} F(\mathbb{N}_3) = F(\mathbb{N}_4) \longrightarrow_{\beta,\beta_\square,ftch} F(\mathbb{N}_5) = F(\mathbb{N}_6) \longrightarrow_{\beta,\beta_\square,ftch} \dots$$

Finally, we arrive at the following infinite reduction sequence (Lem. 7) of typable terms (Lem. 6) in $\lambda^{\mathbf{1},\rightarrow}$, thus contradicting SN of $\lambda^{\mathbf{1},\rightarrow}$:

$$T(F(\mathbb{N}_1)) = T(F(\mathbb{N}_2)) \longrightarrow^+_\beta T(F(\mathbb{N}_3)) = T(F(\mathbb{N}_4)) \longrightarrow^+_\beta T(F(\mathbb{N}_5)) = T(F(\mathbb{N}_6)) \longrightarrow^+_\beta \dots$$

In order to complete our proof we now address our claim, namely that $\longrightarrow 1, 2.1, 4, 7$ reduction is SN. It is the proof of this result that has motivated the modified reduction semantics presented at the beginning of this section. First a simple yet useful result for proving SN of combinations of binary relations that we have implicitly made use of above.

Lemma 8. *Let $\longrightarrow_1$ and $\longrightarrow_2$ be binary relations over some set X. Suppose $\longrightarrow_1$ is SN and $\mathcal{M}$ is a mapping from X to some well-founded set such that:*

1. *$x \longrightarrow_1 y$ implies $\mathcal{M}(x) = \mathcal{M}(y)$*
2. *$x \longrightarrow_2 y$ implies $\mathcal{M}(x) > \mathcal{M}(y)$*

Then $\longrightarrow_1 \cup \longrightarrow_2$ is SN.

Lemma 9. *$\longrightarrow_{1,2.1,4,7}$ reduction is SN.*

Proof. First we prove SN of schemes (1) and (4). Then we conclude by resorting to Lem. 8, introducing a measure $\mathcal{M}_2$ such that:

1. $\mathbb{N} \longrightarrow_{1,4} \mathbb{N}'$ implies $\mathcal{M}_2(\mathbb{N}) = \mathcal{M}_2(\mathbb{N}')$ and
2. $\mathbb{N} \longrightarrow_{2.1,7} \mathbb{N}'$ implies $\mathcal{M}_2(\mathbb{N}) > \mathcal{M}_2(\mathbb{N}')$.

We write $|M|$ and $|k|$ for the size of M and k, respectively. Also, we write $|k, M|$ to abbreviate $|k| + |M|$. Consider the measure $\mathcal{M}_1$ of machine states over pairs of natural numbers (ordered lexicographically):

$$\mathcal{M}_1(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |M| \rangle$$

This measure strictly decreases when schemes (1) and (4) are applied[3].

Measure $\mathcal{M}_2$ is defined as follows:

$$\mathcal{M}_2(\mathbb{W}; w : [k, M]) =_{def} \langle |\mathbb{W}|, |k, M| - len(k) - m(M) \rangle$$

where $len(k)$ is the length of k and m is the following mapping from closed terms to positive integers:

[3] It also decreases when (7) is applied. However, it does not decrease when (2) is applied.

$$\begin{aligned} m(V) &=_{def} 0 \\ m(M\,N) &=_{def} 1 + m(M) \\ m(\mathit{unpack}\ M\ \mathit{to}\ \langle v^{\bullet}, v^{\circ}\rangle\ \mathit{in}\ N) &=_{def} 1 + m(M) \\ m(\mathit{fetch}[w]\,M) &=_{def} 1 \end{aligned}$$

This measure decreases strictly for both (2.1) and (7), whereas it yields equal numbers for (1) and (4).

We can finally state our desired result, whose proof we have presented above.

Proposition 3. $\longrightarrow$ *is SN.*

7 Related Work

There are many foundational calculi for concurrent and distributed programming. Since the focus of this work is on logically motivated such calculi we comment on related work from this viewpoint. To the best of our knowledge, the extant literature does not address calculi for both mobility/concurrency and code certification in a unified theory. Regarding mobility, however, a number of ideas have been put forward. The closest to this article is the work of Moody [Moo04], that of Murphy et al [VCHP04, VCH05, VCH07] and that of Jia and Walker [JW04]. Moody suggests an operational reading of proofs in an intuitionistic fragment of S4 also based on a judgemental analysis of this logic [DP01a]. It takes a step further in terms of obtaining a practical programming language for mobility in that it addresses effectful computation (references and reference update). Also, the diamond connective is considered. Worlds are deliberately left implicit. The author argues this "encourages the programmer to work locally". Murphy et al also introduce a mobility inspired operational interpretation of a Natural Deduction presentation of propositional modal logic, although S5 is considered in their work (both intuitionistic [VCHP04] and classical [VCH05]). They also introduce explicit reference to worlds in their programming model. Operational semantics in terms of abstract machines is considered [VCHP04, VCH05] and also a big-step semantics on terms [Mur08]. Both necessity and possibility modalities are considered. Finally, they explore a type preserving compiler for a prototype language for client/server applications based on their programming model [VCH07]. Jia and Walker [JW04] also present a term assignment for a hybrid modal logic close to S5. They argue that the hybrid approach gives the programmer a tighter control over code distribution. Finally, Borghuis and Feij [BF00] introduce a calculus of stationary services and mobile values whose type system is based on modal logic. Mobility however may not be internalized as a proposition. For example, $\Box^{o}(A \supset B)$ is the type of a service located at o that computes values of B given one of type A. None of the cited works incorporate the notion of certificate in their systems.

8 Conclusion

We present a Curry-de Bruijn-Howard analysis of an intuitionistic fragment (ILP) of the Logic of Proofs LP. We start from a Natural Deduction presentation for

ILP and associate propositions and proofs of this system to types and terms of a mobile calculus $\lambda_{\Box}^{\mathsf{Cert}}$. The modal type constructor $[s]A$ is interpreted as the type of *mobile units*, expressions composed of a code and certificate component. $\lambda_{\Box}^{\mathsf{Cert}}$ has thus language constructs for both code and certificates. Its type system is a unified theory in which both code and certificate construction are verified. Indeed, when mobile units are constructed from the code of other mobile units, the type system verifies not only that the former is mobile in nature (i.e. depends on no local resources) but also that the certificate for this new mobile unit is correctly assembled from the certificates of the latter.

Although we deal exclusively with the necessity modality, we hasten to mention that it would be quite straightforward to add inference schemes for a possibility modality, in the line of related literature (cf. Sec. 7). A term of type $\Diamond A$ is generally interpreted to denote a value of a term at a remote location. However, a provability interpretation of this connective in an intuitionistic fragment of LP has first to be investigated. Since LP is based on classical logic $\Diamond$ is ignored altogether. However, in an intuitionistic setting the interpretation of $\Diamond$ in possible world semantics is not as uncontroversial as that of the necessity modality [Sim94, Ch.3]. Nevertheless one could explore this additional modality from a purely programming languages perspective.

Although $\lambda_{\Box}^{\mathsf{Cert}}$ is meant to be concept-of-proof language, it clearly does not provide the features needed to build extensive examples. Two basic additions that should be considered are references (and computation with effects) and recursion.

References

[AB04] Artëmov, S., Beklemishev, L.: Provability logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 13, pp. 189–360. Kluwer, Dordrecht (2004)

[AB07] Artëmov, S.N., Bonelli, E.: The intensional lambda calculus. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 12–25. Springer, Heidelberg (2007)

[Art95] Artemov, S.: Operational modal logic. Technical Report MSI 95-29, Cornell Univsersity (1995)

[Art01] Artemov, S.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)

[AtC06] Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., Wolter, F., van Benthem, J. (eds.) Handbook of Modal Logics. Elsevier, Amsterdam (2006)

[BF] Bonelli, E., Feller, F.: The logic of proofs as a foundation for certifying mobile computation, `http://www.lifia.info.unlp.edu.ar/~eduardo/lpCertFull.pdf`

[BF00] Borghuis, T., Feijs, L.M.G.: A constructive logic for services and information flow in computer networks. Comput. J. 43(4), 274–289 (2000)

[CH00] Curien, P.-L., Herbelin, H.: The duality of computation. In: ICFP, pp. 233–243 (2000)

[DP01a] Davies, R., Pfenning, F.: A judgmental reconstruction of modal logic. Mathematical Structures in Computer Science 11, 511–540 (2001)

[DP01b] Davies, R., Pfenning, F.: A modal analysis of staged computation. J. ACM 48(3), 555–604 (2001)

[Her94] Herbelin, H.: A lambda-calculus structure isomorphic to gentzen-style sequent calculus structure. In: Pacholski, L., Tiuryn, J. (eds.) CSL 1994. LNCS, vol. 933, pp. 61–75. Springer, Heidelberg (1995)

[JW04] Jia, L., Walker, D.: Modal proofs as distributed programs (extended abstract). In: Schmidt, D. (ed.) ESOP 2004. LNCS, vol. 2986, pp. 219–233. Springer, Heidelberg (2004)

[Moo04] Moody, J.: Logical mobility and locality types. In: Etalle, S. (ed.) LOPSTR 2004. LNCS, vol. 3573, pp. 69–84. Springer, Heidelberg (2005)

[Mur08] Murphy VII, T.: Modal Types for Mobile Code. PhD thesis, Carnegie Mellon (draft) (January 2008)

[Sim94] Simpson, A.: The Proof Theory and Semantics of Intuitionistic Modal Logic. PhD thesis, University of Edinburgh (1994)

[TS97] Taha, W., Sheard, T.: Multi-stage programming. In: ICFP, p. 321 (1997)

[VCH05] Murphy VII, T., Crary, K., Harper, R.: Distributed control flow with classical modal logic. In: Ong, L. (ed.) CSL 2005. LNCS, vol. 3634, pp. 51–69. Springer, Heidelberg (2005)

[VCH07] Murphy VII, T., Crary, K., Harper, R.: Type-safe distributed programming with ml5. In: Barthe, G., Fournet, C. (eds.) TGC 2007 and FODO 2008. LNCS, vol. 4912, pp. 108–123. Springer, Heidelberg (2008)

[VCHP04] Murphy VII, T., Crary, K., Harper, R., Pfenning, F.: A symmetric modal lambda calculus for distributed computing. In: LICS, pp. 286–295. IEEE Computer Society, Los Alamitos (2004)

[WLPD98] Wickline, P., Lee, P., Pfenning, F., Davies, R.: Modal types as staging specifications for run-time code generation. ACM Comput. Surv. 30(3es), 8 (1998)

ATL with Strategy Contexts and Bounded Memory

Thomas Brihaye[1], Arnaud Da Costa[2],
François Laroussinie[3], and Nicolas Markey[2]

[1] Institut de mathématiques, Université de Mons-Hainaut, Belgium
[2] Lab. Spécification & Vérification, ENS Cachan – CNRS UMR 8643, France
[3] LIAFA, Univ. Paris 7 – CNRS UMR 7089, France
thomas.brihaye@umh.ac.be, dacosta@lsv.ens-cachan.fr,
francoisl@liafa.jussieu.fr, markey@lsv.ens-cachan.fr

Abstract. We extend the alternating-time temporal logics ATL and ATL* with *strategy contexts* and *memory constraints*: the first extension makes strategy quantifiers to not "forget" the strategies being executed by the other players. The second extension allows strategy quantifiers to restrict to memoryless or bounded-memory strategies.

We first consider expressiveness issues. We show that our logics can express important properties such as equilibria, and we formally compare them with other similar formalisms (ATL, ATL* , Game Logic, Strategy Logic, ...). We then address the problem of model-checking for our logics, especially we provide a PSPACE algorithm for the sublogics involving only memoryless strategies and an EXPSPACE algorithm for the bounded-memory case.

1 Introduction

Temporal logics and model checking. Temporal logics (LTL, CTL) have been proposed for the specification of reactive systems almost thirty years ago [13,7,14]. Since then, they have been widely studied and successfully used in many situations, especially for *model checking*—the automatic verification that a model of a system satisfies a temporal logic specification.

Alternating-time temporal logic (ATL). Over the last ten years, ATL has been proposed as a new flavor of temporal logics for specifying and verifying properties in *multi-agent systems* (modeled as *Concurrent Game Structures* (CGS) [2]), in which several agents can concurrently act upon the behaviour of the system. In these models, it is not only interesting to know if something *can* or *will* happen, as is expressed in CTL or LTL, but also if some agent(s) can *control* the evolution of the system in order to enforce a given property, whatever the other agents do. ATL can express this kind of properties thanks to its quantifier over strategies, denoted $\langle\!\langle A \rangle\!\rangle$ (where A is a coalition of agents). That coalition A has a strategy for reaching a winning location is then written $\langle\!\langle A \rangle\!\rangle \mathbf{F}\, \mathtt{win}$ (where $\mathbf{F}$ is the LTL modality for "eventually").

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 92–106, 2009.

Our contributions. In this paper, we extend ATL and ATL* in two directions: first, while ATL strategy quantifiers drop strategies introduced by earlier quantifiers in the evaluation of the formula, our logics keep executing those strategies. To achieve this idea, we naturally adapt the semantics of ATL* in order to interpret a formula within a *stategy context*. Our new strategy quantifier, written $\langle\!\langle A \rangle\!\rangle$, can for instance express that "A has a strategy s.t. (1) Player B always has a strategy (given that of A) to enforce Φ and (2) Player C always has a strategy (given the *same* strategy of A) to enforce Ψ". This would be written as follows: $\langle\!\langle A \rangle\!\rangle \mathbf{G} \left(\langle\!\langle B \rangle\!\rangle \Phi \wedge \langle\!\langle C \rangle\!\rangle \Psi \right)$. Naive attempts to express this property in standard ATL fail: in the ATL formula $\langle\!\langle A \rangle\!\rangle \mathbf{G} \left(\langle\!\langle B \rangle\!\rangle \Phi \wedge \langle\!\langle C \rangle\!\rangle \Psi \right)$, the coalitions do not cooperate anymore; in $\langle\!\langle A \rangle\!\rangle \mathbf{G} \left(\langle\!\langle A, B \rangle\!\rangle \Phi \wedge \langle\!\langle A, C \rangle\!\rangle \Psi \right)$, coalition A is allowed to use different strategies when playing with B and C.

Our second extension consists in parameterising strategy quantifiers with the *resources* (in terms of memory) allowed for strategies: we define the quantifier $\langle\!\langle A^s \rangle\!\rangle$ with $s \in (\mathbb{N} \cup \{\infty\})$, which restricts the quantification to strategies using memory of size s (called s-memory strategies) for Player A. It is well-known that memoryless strategies are enough to enforce ATL properties, but this is not the case for ATL* formulae, nor for our extension of ATL (and ATL*) with strategy contexts.

Our results are twofold: on the one hand, we study the increase in expressiveness brought by our extensions, comparing our logics to ATL and ATL* and several related logics such as Game Logic [2], Strategy Logic [6] and $\mathrm{Q}D_\mu$ [12], ... We also illustrate their convenience with some sample formulas expressing *e.g.* equilibrium properties.

On the other hand, we study the model-checking problem for our extensions: while we only have a non-elementary algorithm for the most general logic, we propose a polynomial-space algorithm for model-checking our logic in the memoryless case, and extend it to an exponential-space algorithm for the bounded-memory setting.

Related work. Recently, several works have focused on the same kind of extensions of ATL, and come up with different solutions which we list below. Generally speaking, this leads to very expressive logics, able to express for instance equilibrium properties, and drastically increases the model-checking complexity.

- IATL [1] extends ATL with strategy contexts, with a similar definition as ours, but it requires players to commit to a strategy, which they are not allowed to modify in the sequel. This logic is then studied in the memoryless case (which is proven to be a strict restriction to memory-based strategies).
- SL [6] extends temporal logics with first-order quantification over strategies. This extension has been defined and studied only in the two-player turn-based setting, where a non-elementary algorithm is proposed.
- $\mathrm{Q}D_\mu$ [12] considers strategies as labellings of the computation tree of the game structure with fresh atomic propositions. This provides a way of explicitly dealing with strategies. This extension is added on top of the decision μ-calculus $D\mu$, yielding a very expressive, yet decidable framework.

– Stochastic Game Logic [3] is a similar extension to ours, but for stochastic games. It is undecidable in the general case, but proved decidable when restricting to memoryless strategies.

Instead of defining a completely new formalism, we prefer sticking to an ATL-like syntax, as we believe that our new modality $\langle\!\langle \cdot \rangle\!\rangle$ is more intuitive than the standard ATL modality $\langle\!\langle A \rangle\!\rangle$. Also, none of the above the extension has the ability to explicitely restrict to bounded-memory strategies, which is of obvious practical relevance and leads to more efficient algorithms.

Plan of the paper. Section 2 contains the definitions of our logics, and of our bounded-memory setting. Section 3 deals with the expressiveness results, and compares our extension with those cited in the related work above. In Section 4, we consider the model-checking problem for our extensions, and provide algorithms for the case of s-memory strategies. For lack of space, we refer to the full version [4] of this paper for the detailled proofs.

2 Definitions

In this section we introduce classical definitions of concurrent game structures, strategies and outcomes. We then define a notion of s-bounded memory strategies. In the whole paper, AP denotes a finite non-empty set of atomic propositions.

2.1 Concurrent Game Structures

Concurrent game structures are a multi-player extension of classical Kripke structures [2]. Their definition is as follows:

Definition 1. *A* Concurrent Game Structure *(*CGS *for short)* $\mathcal{C}$ *is a* 8*-tuple* $(\mathsf{Loc}, \ell_0, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edg})$ *where:*

- Loc *is a finite set of* locations, $\ell_0 \in \mathsf{Loc}$ *is the initial location;*
- $\mathsf{Lab}\colon \mathsf{Loc} \to 2^{AP}$ *is a labelling function;*
- $\delta \subseteq \mathsf{Loc} \times \mathsf{Loc}$ *is the set of transitions;*
- $\mathsf{Agt} = \{A_1, ..., A_k\}$ *is a finite set of* agents *(or* players*);*
- $\mathcal{M}$ *is a finite, non-empty set of moves;*
- $\mathsf{Mov}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \setminus \{\varnothing\}$ *defines the (finite) set of possible moves of each agent in each location.*
- $\mathsf{Edg}\colon \mathsf{Loc} \times \mathcal{M}^k \to \delta$*, where* $k = |\mathsf{Agt}|$*, is a transition table. With each location and each set of moves of the agents, it associates the resulting transition.*

The size $|\mathcal{C}|$ of a CGS $\mathcal{C}$ is defined as $|\mathsf{Loc}| + |\mathsf{Edg}|$, where $|\mathsf{Edg}|$ is the size of the transition table[1]. The intended behaviour is as follows [2]: in a location ℓ,

[1] Our results would still hold if we consider symbolic CGSs [10], where the transition table is encoded through boolean formulas.

each player A_i chooses one of his possible moves m_{A_i} and the next transition is given by $\mathsf{Edg}(\ell, m_{A_1}, ..., m_{A_k})$. We write $\mathsf{Next}(\ell)$ for the set of all transitions corresponding to possible moves from ℓ, and $\mathsf{Next}(\ell, A_j, m)$, with $m \in \mathsf{Mov}(\ell, A_j)$, for the restriction of $\mathsf{Next}(\ell)$ to possible transitions from ℓ when player A_j makes the move m.

2.2 Coalitions, Bounded-Memory Strategies, Outcomes

Coalitions. A coalition is a subset of agents. In multi-agent systems, a coalition A plays against its opponent coalition $\mathsf{Agt} \setminus A$ as if they were two single players. We thus extend Mov and Next to coalitions:

- Given $A \subseteq \mathsf{Agt}$ and $\ell \in \mathsf{Loc}$, $\mathsf{Mov}(\ell, A)$ denotes the set of possible moves for coalition A from ℓ. Those moves m are composed of one single move per agent of the coalition, *i.e.*, $m = (m_a)_{a \in A}$.
- Next is extended to coalitions in a natural way: given $m = (m_a)_{a \in A} \in \mathsf{Mov}(\ell, A)$, we let $\mathsf{Next}(\ell, A, m)$ denote the restriction of $\mathsf{Next}(\ell)$ to locations reachable from ℓ when every player $A_j \in A$ makes the move m_{A_j}.

Strategies and outcomes. Let $\mathcal{C}$ be a CGS. A *computation* of $\mathcal{C}$ is an infinite sequence $\rho = \ell_0 \ell_1 \ldots$ of locations such that for any i, $\ell_{i+1} \in \mathsf{Next}(\ell_i)$. We write ρ^i for the i-th suffix of ρ, and $\rho[i...j]$ for part of ρ between ℓ_i and ℓ_j. In particular, $\rho[i]$ denotes the $i+1$-st location ℓ_i. A *strategy* for a player $A_i \in \mathsf{Agt}$ is a function f_{A_i} that maps any finite prefix of a computation to a possible move for A_i, *i.e.*, satisfying $f_{A_i}(\ell_0 \ldots \ell_m) \in \mathsf{Mov}(\ell_m, A_i)$. A strategy is *memoryless* if it only depends on the current state (*i.e.*, $f_{A_i}(\ell_0 \ldots \ell_m) = f_{A_i}(\ell_m)$). A strategy for a coalition A of agents is a set of strategies, one for each agent in the coalition. The set of strategies (resp. memoryless strategies) for A is denoted $\mathsf{Strat}(A)$ (resp. $\mathsf{Strat}^0(A)$).

A strategy for A_j induces a set of computations from ℓ, called the *outcomes* of f_{A_j} from ℓ and denoted $\mathsf{Out}(\ell, f_{A_j})$, that player A_j can enforce: $\ell_0 \ell_1 \ldots \in \mathsf{Out}(\ell, f_{A_j})$ iff $\ell_0 = \ell$ and $\ell_{i+1} \in \mathsf{Next}(\ell_i, A_j, f_{A_j}(\ell_0 \ldots \ell_i))$ for any i. Given a coalition A, a strategy for A is a tuple F_A containing one strategy for each player in A: $F_A = \{f_{A_j} | A_j \in A\}$. The *domain* of F_A ($\mathsf{dom}(F_A)$) is A. The strategy f_{A_j} for A_j is also denoted $(F_A)_{|A_j}$; more generally, $(F_A)_{|B}$ (resp. $(F_A)_{\setminus B}$) denotes the restriction of F_A to the coalition $A \cap B$ (resp. $A \setminus B$). The outcomes of F_A from a location ℓ are the computations enforced by the strategies in F_A: $\ell_0 \ell_1 \ldots \in \mathsf{Out}(\ell, F_A)$ iff $\ell_0 = \ell$ and for any i, $\ell_{i+1} \in \mathsf{Next}(\ell_i, A, (f_{A_j}(\ell_0, \ldots, \ell_i))_{A_j \in A})$. Note that $\mathsf{Out}(\ell, F_A) \subseteq \mathsf{Out}(\ell, (F_A)_{|B})$ for any coalitions A and B, and in particular that $\mathsf{Out}(\ell, F_\varnothing)$ represents the set of all computations from ℓ.

It is also possible to *combine* two strategies $F \in \mathsf{Strat}(A)$ and $F' \in \mathsf{Strat}(B)$, resulting in a strategy $F \circ F' \in \mathsf{Strat}(A \cup B)$ defined as follows: $(F \circ F')_{|A_j}(\ell_0 \ldots \ell_m)$ equals $F_{|A_j}(\ell_0 \ldots \ell_m)$ if $A_j \in A$, and it equals $F'_{|A_j}(\ell_0 \ldots \ell_m)$ if $A_j \in B \setminus A$.

Finally, given a strategy F, an execution ρ and some integer $i \geq 0$, we define the strategy $F^{\rho,i}$ corresponding to the behaviour of F after prefix $\rho[0...i]$

as follows: $F^{\rho,i}(\pi) = F(\rho[0\dots i] \cdot \pi)$. Note that if F is memoryless, then $F^{\rho,i} = F$.

Bounded-memory strategies. Between general strategies (without bound over its resources) and *memoryless* strategies, we can consider *bounded-memory strategies*. Let s be a (binary-encoded) integer representing the size of the memory. We define a bounded memory strategy as a memoryless strategy over the locations of the CGS **and** a set of memory cells [11,16]: choosing the move depends on both the location and the current memory cell, and after every move, the player can "update" its memory by moving to another cell. The size of the memory is then defined as the number of cells. Let Cell be the set of $s+1$ memory cells $\{0,\dots,s\}$.

Formally an s-memory strategy F_A for Player A is a 3-tuple $(F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$ where: F^{mov} is a mapping from $\mathsf{Cell} \times \mathsf{Loc}$ to $\mathcal{M}$ that associates a move with the current memory cell and the current location of the CGS, F^{cell} is a mapping from $\mathsf{Cell} \times \mathsf{Loc}$ to Cell that updates the memory cell, and c is the current memory cell of this strategy. For the sake of readability, given a bounded-memory strategy $F_A = (F^{\mathsf{mov}}, F^{\mathsf{cell}}, c)$, we still write $F_A(\ell)$ for $F^{\mathsf{mov}}(c, \ell)$.

The notions of computations and outcomes are easily extended to this new setting: the set $\mathsf{Next}(\ell, A, F_A(\ell))$ contains the possible successor locations when A plays from ℓ according to F_A. Of course, the memory cell of F_A changes along an execution ρ, and we define $F_A^{\rho,i}$ as the strategy $(F^{\mathsf{mov}}, F^{\mathsf{cell}}, c_i)$ where c_i is defined inductively with: $c_0 = c$ and $c_{j+1} = F^{\mathsf{cell}}(\rho[j], c_j)$. Finally the outcomes $\mathsf{Out}(\ell, F_A)$ are the executions $\rho = \ell_0\ell_1\dots$ such that $\ell_{j+1} \in \mathsf{Next}(\ell_j, A, F_A^{\rho,j}(\ell_j))$.

Coalitions are handled the usual way: we use pairs $(A, \overline{s})$ to represent a coalition $A \subseteq \mathsf{Agt}$ and a memory-bounds vector $\overline{s} \in (\mathbb{N} \cup \{\infty\})^A$ which associates a size $\overline{s}(A_j)$ with the memory that agent $A_j \in A$ can use for its strategy. The set of strategies for A with memory bound $\overline{s}$ is denoted $\mathsf{Strat}^{\overline{s}}(A)$, and we omit to mention the memory bound when none is imposed.

2.3 The Logic $\mathsf{ATL}^\star_{sc,\infty}$

We now define the logic $\mathsf{ATL}^\star_{sc,\infty}$ that extends $\mathsf{ATL}^\star$ with strategy contexts and bounded-memory strategy quantifiers:

Definition 2. *The syntax of $\mathsf{ATL}^\star_{sc,\infty}$ is defined by the following grammar:*

$$\mathsf{ATL}^\star_{sc,\infty} \ni \varphi_s, \psi_s ::= P \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle\!\langle A, \overline{s}\rangle\!\rangle\, \varphi_p \mid [\![A]\!]\, \varphi_s$$
$$\varphi_p, \psi_p ::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\, \mathbf{U}\, \psi_p$$

with $P \in \mathsf{AP}$, $A \subseteq \mathsf{Agt}$ and $\overline{s} \in (\mathbb{N} \cup \{\infty\})^A$. Formulas defined as φ_s are called state formulas*, while φ_p defines* path formulas.

An $\mathsf{ATL}^\star_{sc,\infty}$ formula Φ is interpreted over a state ℓ of a CGS $\mathcal{C}$ *within a strategy context* $F \in \mathsf{Strat}(B)$ *for some coalition* B; this is denoted by $\ell \models_F \Phi$. The semantics is defined as follows:

$$
\begin{array}{rll}
\ell \models_F \langle\!\cdot A, \overline{s} \cdot\rangle\!\rangle\, \varphi_p & \text{iff} & \exists F_A \in \mathsf{Strat}^{\overline{s}}(A).\ \forall \rho \in \mathsf{Out}(\ell, F_A \circ F).\ \rho \models_{F_A \circ F} \varphi_p, \\
\ell \models_F \rangle\!\cdot A \cdot\langle\!\rangle\, \varphi_s & \text{iff} & \ell \models_{F_{\setminus A}} \varphi_s, \\
\rho \models_F \varphi_s & \text{iff} & \rho[0] \models_F \varphi_s, \\
\rho \models_F \mathbf{X}\, \varphi_p & \text{iff} & \rho^1 \models_{F^{\rho,1}} \varphi_p, \\
\rho \models_F \varphi_p\, \mathbf{U}\, \psi_p & \text{iff} & \exists i.\ \rho^i \models_{F^{\rho,i}} \psi_p \text{ and } \forall 0 \leq j < i.\ \rho^j \models_{F^{\rho,j}} \varphi_p.
\end{array}
$$

Given a CGS $\mathcal{C}$ with initial location ℓ_0, and an $\mathsf{ATL}^\star_{sc,\infty}$ formula Φ, the model-checking problem consists in deciding whether[2] $\ell_0 \models_\varnothing \Phi$.

The formula $\langle\!\cdot A, \overline{s} \cdot\rangle\!\rangle\, \varphi$ holds on a location ℓ within a context F for a coalition B iff there exists a $\overline{s}$-memory strategy for A to enforce φ when B plays according to the strategy F. We use $\langle\!\cdot A \cdot\rangle\!\rangle$ to denote the modality with no restriction over the memory allowed for the strategies of A (*i.e.*, the modality $\langle\!\cdot A, \infty^A \cdot\rangle\!\rangle$); and we use $\langle\!\cdot A^0 \cdot\rangle\!\rangle$ as an abbreviation for $\langle\!\cdot A, 0^A \cdot\rangle\!\rangle$ to consider only memoryless strategies.

Conversely the modality $\rangle\!\cdot A \cdot\langle\!\rangle$ removes the strategy for A from the current context under which the formula is interpreted. The operator $\rangle\!\cdot \mathsf{Agt} \cdot\langle\!\rangle$ allows us to empty the current context, and then we clearly have: $\ell \models_F \rangle\!\cdot \mathsf{Agt} \cdot\langle\!\rangle\, \varphi \Leftrightarrow \ell \models_{F'} \rangle\!\cdot \mathsf{Agt} \cdot\langle\!\rangle\, \varphi$ for any context F and F'.

This entails that $\mathsf{ATL}^\star_{sc,\infty}$ contains $\mathsf{ATL}^\star$ (thus also $\mathsf{CTL}^\star$). Indeed the classical strategy quantifier of $\mathsf{ATL}^\star$, namely $\langle\!\langle A \rangle\!\rangle$, does not handle strategy contexts: $\langle\!\langle A \rangle\!\rangle\, \varphi$ holds for a location ℓ iff A has a strategy to enforce φ whatever the choices of $\mathsf{Agt} \setminus A$. Clearly $\langle\!\langle A \rangle\!\rangle\, \varphi$ is equivalent to $\rangle\!\cdot \mathsf{Agt} \cdot\langle\!\rangle\, \langle\!\cdot A \cdot\rangle\!\rangle\, \varphi$.

Obviously the existence of an $\overline{s}$-memory strategy for A to enforce φ entails the existence of an $\overline{s'}$-memory strategy if $\overline{s'} \geq \overline{s}$ (*i.e.*, $\overline{s'}(A_j) \geq \overline{s}(A_j)$ for all $A_j \in A$). Note that the converse is not true except for special cases such as ATL where memoryless strategies are sufficient (see [2,15]).

We will use standard abbreviations such as $\top = P \vee \neg P$, $\bot = \neg\top$, $\mathbf{F}\, \varphi = \top\, \mathbf{U}\, \varphi$, etc.

Now we introduce several fragments of $\mathsf{ATL}^\star_{sc,\infty}$:

- $\mathsf{ATL}^\star_{sc,b}$ (with $b \in \mathbb{N}$) is the fragment of $\mathsf{ATL}^\star_{sc,\infty}$ where the quantifiers $\langle\!\cdot A, \overline{s} \cdot\rangle\!\rangle$ only use memory-bounds less than or equal to b. In particular, $\mathsf{ATL}^\star_{sc,0}$ only allows memoryless strategies.
- $\mathsf{ATL}^\star_{sc}$ is the fragment of $\mathsf{ATL}^\star_{sc,\infty}$ where no restriction over the memory is allowed (any strategy quantifier deals with infinite-memory strategies).
- $\mathsf{ATL}_{sc,\infty}$ contains the formulae where every temporal modality is in the immediate scope of a strategy quantifier (*i.e.*, the path formulae are restricted to $\varphi_s\, \mathbf{U}\, \psi_s$, $\varphi_s\, \mathbf{R}\, \psi_s$—$\mathbf{R}$ is the "dual-until" modality—and $\mathbf{X}\, \varphi_s$). It follows from the above assertion that $\mathsf{ATL}_{sc,\infty}$ contains ATL and CTL. We also define the fragments $\mathsf{ATL}_{sc,b}$ and ATL_{sc} as above.

[2] The context can be omitted when it is empty, and we can directly write $\ell \models \Phi$.

3 Expressiveness

In this section, we consider expressiveness issues, first illustrating the ability of $\mathsf{ATL}^{\star}_{sc,\infty}$ to state interesting properties, and then comparing it with related formalisms.

3.1 Some Interesting Formulas of $\mathsf{ATL}^{\star}_{sc,\infty}$

The new modalities $\langle\cdot A\rangle$ allow us to express many interesting properties over the strategies of different players in a game. In [4], we show how our logics can express the different properties that motivated the introduction of SL, $\mathrm{Q}D_{\mu}$ or IATL. Here we just give a few examples.

Nash equilibria. Given two players A_1 and A_2 having their own objectives Φ_1 and Φ_2, two strategies F_1 and F_2 for players 1 and 2 respectively form a *Nash equilibrium* if there is no "better" strategy F_1' for A_1 w.r.t. Φ_1 when Player 2 plays according to F_2, and *vice versa*. Given a strategy context $F = (F_1, F_2)$, the following formula holds in state ℓ under F iff F_1 and F_2 form a Nash equilibrium in ℓ:

$$\Big((\langle\cdot A_1\rangle \Phi_1) \Rightarrow \Phi_1 \ \wedge \ (\langle\cdot A_2\rangle \Phi_2) \Rightarrow \Phi_2 \Big)$$

This provides us with a way of expressing the existence of Nash equilibria having extra properties.

Winning secure equilibria. The *winning secure equilibrium* [5] (WSE) is a stronger notion of equilibrium: two strategies F_1 and F_2, for players 1 and 2 with objectives Φ_1 and Φ_2 respectively, form a WSE if each player has no better strategy for himself, and no worse strategy for his opponent. Again, the strategy context F is a winning secure equilibrium in ℓ iff the following formula holds in ℓ within F:

$$\begin{aligned}&(\langle\cdot A_1\rangle \Phi_1) \Rightarrow \Phi_1 \ \wedge \ (\langle\cdot A_2\rangle \Phi_2) \Rightarrow \Phi_2 \ \wedge \\ &\qquad \Big(\langle\cdot A_1\rangle (\Phi_1 \wedge \neg\Phi_2) \Rightarrow (\Phi_1 \wedge \neg\Phi_2) \Big) \ \wedge \ \Big(\langle\cdot A_2\rangle (\Phi_2 \wedge \neg\Phi_1) \Rightarrow (\Phi_2 \wedge \neg\Phi_1) \Big)\end{aligned}$$

Client-server interactions. Given a protocol where a server S has to treat the requests of differents agents A_1, ..., A_n, we can express that S has a strategy to ensure that every agent A_i can act in order to make its requests to be granted. Such a property can be stated as folows:

$$\langle\cdot S\rangle \, \mathbf{G} \Big[\bigwedge_{i=1\ldots n} \Big(\mathsf{req}_i \Rightarrow \langle\cdot A_i\rangle \, \mathbf{F} \, \mathsf{grant}_i \Big) \Big]$$

Clearly this property requires the use of strategy contexts because every agent has to cooperate with the server (but not with other agents).

3.2 Expressiveness of $\cdot\rangle A\langle\cdot$ Quantifier

We have illustrated the use of modality $\cdot\rangle A\langle\cdot$ by expressing the classical $\mathsf{ATL}^\star$ modality $\langle\!\langle A\rangle\!\rangle$ with $\cdot\rangle\mathsf{Agt}\langle\cdot\ \langle\cdot A\cdot\rangle\cdot$: we first forget the current strategy context and then quantify over the existence of a strategy for A: relaxing is necessary because it has to be a real strategy, *i.e.*, correct for any choice for the other agents. In fact, this modality does not add expressive power to $\mathsf{ATL}^\star_{sc,\infty}$:

Proposition 3. *For any $\mathsf{ATL}^\star_{sc,\infty}$ formula Φ, there exists a formula Ψ containing no $\cdot\rangle\cdot\langle\cdot$ modality such that $\Phi \equiv \Psi$.*

Proof. Given a subset of agents $C \subseteq \mathsf{Agt}$ and $\Phi \in \mathsf{ATL}^\star_{sc,\infty}$, we define formula $\overline{\Phi}^C$ recursively as follows (in this definition, $[\cdot C\cdot]\,\varphi \stackrel{\text{def}}{=} \neg\langle\cdot C\cdot\rangle\neg\varphi$):

$$\overline{\langle\cdot A,\overline{s}\cdot\rangle\,\Phi}^C \stackrel{\text{def}}{=} \langle\cdot A,\overline{s}\cdot\rangle\,[\cdot C\setminus A\cdot]\,\overline{\Phi}^{C\setminus A} \qquad \overline{\cdot\rangle A\langle\cdot\,\Phi}^C \stackrel{\text{def}}{=} \overline{\Phi}^{C\cup A}$$

$$\overline{\Phi\,\mathbf{U}\,\Psi}^C \stackrel{\text{def}}{=} \overline{\Phi}^C\,\mathbf{U}\,\overline{\Psi}^C \qquad \overline{\mathbf{X}\,\Phi}^C \stackrel{\text{def}}{=} \mathbf{X}\,\overline{\Phi}^C$$

$$\overline{\Phi\wedge\Psi}^C \stackrel{\text{def}}{=} \overline{\Phi}^C\wedge\overline{\Psi}^C \qquad \overline{\neg\Phi}^C \stackrel{\text{def}}{=} \neg\overline{\Phi}^C$$

$$\overline{P}^C \stackrel{\text{def}}{=} P$$

Now we have the following lemma:

Lemma 4. *For any strategy context F, any subset $C \subseteq \mathsf{dom}(F)$ and any formula $\Phi \in \mathsf{ATL}^\star_{sc,\infty}$ and any path formula Φ_p, we have:*

$$\begin{array}{rcl} \ell \models_{F_{\setminus C}} \Phi & \Leftrightarrow & \ell \models_F \overline{\Phi}^C \\ \rho \models_{F_{\setminus C}} \Phi_p & \Leftrightarrow & \rho \models_F \overline{\Phi_p}^C \end{array}$$

Proof. The proof is done by structural induction over the formula. In this proof we will use C' as an abbreviation for the coalition $C \setminus A$. Moreover F_B ranges over strategies for coalition B.

– $\Psi \stackrel{\text{def}}{=} \langle\cdot A,\overline{s}\cdot\rangle\,\varphi$.

We have the following equivalences : $\ell \models_F \langle\cdot A,\overline{s}\cdot\rangle\,[\cdot C'\cdot]\,\overline{\varphi}^{C'}$ means by definition

$$\exists F_A.\ \forall F_{C'}.\ \forall\rho \in \mathsf{Out}(q, F_{C'}\circ F_A\circ F).\ \rho \models_{F_{C'}\circ F_A\circ F} \overline{\varphi}^{C'},$$

Then the induction hypothesis yields

$$\exists F_A.\ \forall F_{C'}.\ \forall\rho \in \mathsf{Out}(q, F_{C'}\circ F_A\circ F).\ \rho \models_{(F_{C'}\circ F_A\circ F)_{\setminus(C')}} \varphi,$$

or equivalently, since $(F_{C'}\circ F_A\circ F)_{\setminus C'} = F_A\circ(F_{\setminus C})$:

$$\exists F_A.\ \forall F_{C'}.\ \forall\rho \in \mathsf{Out}(q, F_{C'}\circ F_A\circ F).\ \rho \models_{F_A\circ(F_{\setminus C})} \varphi,$$

or also

$$\exists F_A.\ \forall\rho \in \mathsf{Out}(q, F_A\circ(F_{\setminus C})).\ \rho \models_{(F_A\circ(F_{\setminus C}))} \varphi,$$

since we have

$$\bigcup_{F_{C'} \in \mathsf{Strat}(C')} \mathsf{Out}(q, F_{C'} \circ F_A \circ F) = \mathsf{Out}(q, F_A \circ (F_{\setminus C})).$$

This leads to $\ell \models_{F_{\setminus C}} \langle\!\langle A, \overline{s} \rangle\!\rangle \varphi$, which is the desired result.

– $\Psi \stackrel{\text{def}}{=} \rangle\!\rangle A \langle\!\langle \varphi$. On the one hand, by the semantics of $\mathsf{ATL}^{\star}_{sc,\infty}$, we have that:

$$q\ell \models_{F_{\setminus C}} \rangle\!\rangle A \langle\!\langle \varphi \quad \text{iff} \quad \ell \models_{F_{\setminus (C \cup A)}} \varphi.$$

On the other hand, the induction hypothesis tells us that:

$$\ell \models_{F_{\setminus (C \cup A)}} \varphi \quad \text{iff} \quad \ell \models_F \overline{\varphi}^{C \cup A}.$$

Gathering the two equivalences, we obtain the desired result.

– $\Psi \stackrel{\text{def}}{=} \varphi \,\mathbf{U}\, \psi$. The semantics of $\mathsf{ATL}^{\star}_{sc,\infty}$ tells us that $\rho \models_{F_{\setminus C}} \Psi$ if and only if the following formal holds:

$$\exists i.\ \rho^i \models_{(F_{\setminus C})^{\rho,i}} \psi \text{ and } \forall 0 \leq j < i.\ \rho^j \models_{(F_{\setminus C})^{\rho,j}} \varphi.$$

By using the induction hypothesis, the above formula is equivalent to the following one:

$$\exists i.\ \rho^i \models_{F^{\rho,i}} \overline{\psi}^C \text{ and } \forall 0 \leq j < i.\ \rho^j \models_{F^{\rho,j}} \overline{\varphi}^C,$$

which means that $\rho \models_F \overline{\varphi}^C \,\mathbf{U}\, \overline{\psi}^C$. We thus obtain the desired result.

– The remaining cases are straightforward.

We can now finish the proof by considerng $\Psi = \overline{\Phi}^{\varnothing}$. □

3.3 Comparison with Other Formalisms

Figure 1 summarizes the expressiveness results for our logics. An arrow $L \to L'$ denotes that $L \leq_{ex} L'$, *i.e.*, that L' is at least as expressive as L (*i.e.*, for any formula in L, there exists an equivalent[3] formula in L'). Note that in some cases, the relation is strict and we have $L < L'$, see the corresponding theorems for more details. The dotted arrows correspond to results proved in (the long version of) this paper; plain arrows correspond to literature results (they are labeled with bibliographic references) or direct syntactic inclusions.

The results about ATL, $\mathsf{ATL}^{\star}$, $\mathsf{CTL}^{\star}$ and AMC are presented in [4]: most of them are based on the ability of the new modalities with strategy contexts and/or memory bounds to *distinguish* models that are alternating-bisimilar (and thus satisfy the same formulas of the classical AMC fragments). The full version also contains the proof that adding the quantification over bounded memory increases the expressive power of ATL_{sc} and $\mathsf{ATL}^{\star}_{sc}$.

Here we only develop our results concerning Game Logic, which is a powerful logic to handle properties over strategies, and we discuss the case of Strategy Logic.

[3] That is, having the same truth value in any location of any CGS under any context.

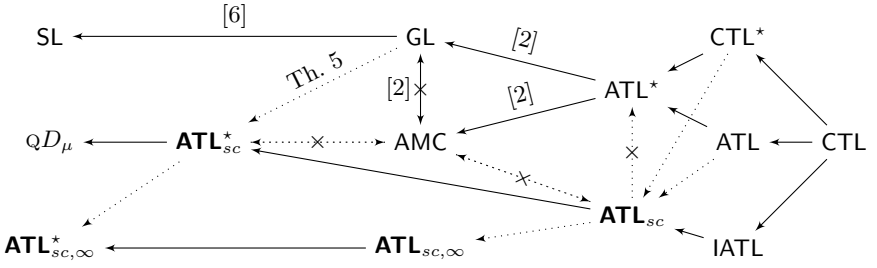


Fig. 1. Expressiveness of $\mathsf{ATL}_{sc,\infty}$ and $\mathsf{ATL}^{\star}_{sc,\infty}$ compared to classical logics

Comparison with Game Logic. *Game Logic* was introduced in [2] in order to express the module-checking problem [9]. This logic is an extension of $\mathsf{ATL}^{\star}$ where the existence of a strategy for A is written $\exists A$, and where it is possible to deal explicitly with the execution tree induced by a strategy: given such a tree t, it is possible to quantify (with modalities $\exists$ and $\forall$) over the executions inside t and specify temporal properties. For example, the formula $\exists A.((\exists P \,\mathbf{U}\, P') \wedge (\forall \mathbf{F}\, P''))$ specifies the existence of a strategy F_A for A s.t. in the tree induced by F_A, we have: (1) there exists a run along which $P \,\mathbf{U}\, P'$ holds and (2) every run satisfies $\mathbf{F}\, P''$. We have the following result:

Theorem 5. *$\mathsf{ATL}^{\star}_{sc}$ $>_{ex}$ GL*

Proof (sketch). First we give a translation from GL into $\mathsf{ATL}^{\star}_{sc}$; given a GL formula φ, we inductively define $\overline{\varphi}$:

$$\overline{\exists A\varphi} \stackrel{\text{def}}{=} \langle\!\langle A \rangle\!\rangle\, \overline{\varphi} \qquad \overline{\exists \varphi} \stackrel{\text{def}}{=} \neg \langle \cdot \varnothing \cdot \rangle \neg \overline{\varphi} \qquad \overline{P} \stackrel{\text{def}}{=} P.$$

The other inductive rules are defined in the natural way. Note that if φ is a GL tree-formula, then $\overline{\varphi}$ is an $\mathsf{ATL}^{\star}_{sc}$ state-formula. In this translation, we use a strategy context to represent the tree used to interpret GL path- and tree-formulae. In the following, given a state ℓ of a CGS and a strategy F for some coalition A, we use $\mathsf{ExecTree}(\ell, F)$ to denote the subtree of the computation tree from ℓ whose infinite rooted paths are the elements of $\mathsf{Out}(\ell, F)$. We must show that for any GL path (resp. tree) formula φ_p (resp. φ_t), any path ρ in some CGS and any strategy F for some coalition A, we have: $(\mathsf{ExecTree}(\rho[0], F), \rho) \models \varphi_p$ iff $\rho \models_F \overline{\varphi_p}$ and $\mathsf{ExecTree}(\rho[0], F) \models \varphi_t$ iff $\rho[0] \models_F \overline{\varphi_t}$. Here we just consider the first equivalence (the second one can be treated in a similar way). The usual induction steps are straightforward, thus we only consider the following two cases:

- $\varphi = \exists A.\psi$. Then $\rho \models_F \overline{\varphi}$ means that there is a strategy F' for coalition A, s.t. any computation ρ' in $\mathsf{Out}(\rho[0], F')$ satisfies $\overline{\psi}$. By i.h., this is equivalent to $\exists F' \in \mathsf{Strat}(A).\ \forall \rho' \in \mathsf{Out}(\rho[0], F').\ (\mathsf{ExecTree}(\rho[0], F'), \rho') \models \psi$, hence to $\rho[0] \models \exists A.\psi$ because ψ is a tree formula. Now $\exists A.\psi$ is a state formula that can be interpreted over any execution tree with root $\rho[0]$, in particular over $\mathsf{ExecTree}(\rho[0], F)$.

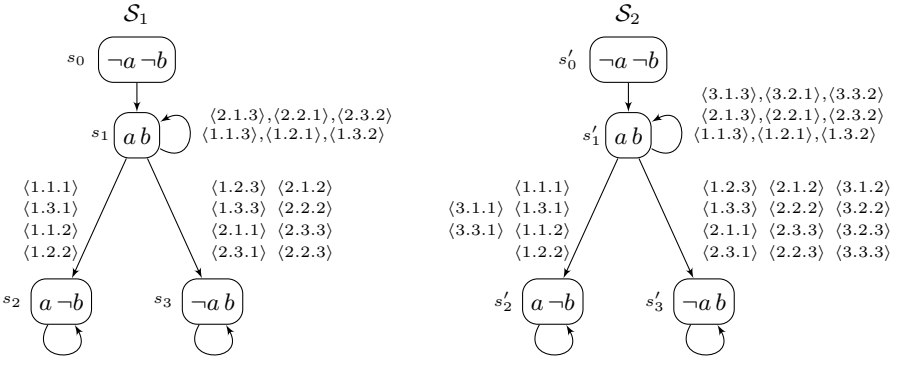


Fig. 2. $\mathcal{S}_1$ and $\mathcal{S}_2$ cannot be distinguished by GL

– $\varphi = \exists\psi$. Then $\rho \models_F \overline{\varphi}$ means that "not all the computations from $\rho[0]$ and following the strategy context F do not satisfy $\overline{\psi}$", and is then equivalent to $\exists \rho' \in \mathsf{Out}(\rho[0], F)$. $\rho' \models_F \overline{\psi}$. Again from the i.h. we obtain the existence of a path in $\mathsf{ExecTree}(\rho[0], F)$ satisfying ψ, and then $\mathsf{ExecTree}(\rho[0], F) \models \exists\psi$, which is equivalent to $(\mathsf{ExecTree}(\rho[0], F), \rho) \models \varphi$ (as φ is a tree formula).

Finally if we consider the case where φ is a state formula and F is the empty strategy, we get that $\overline{\varphi}$ is an $\mathsf{ATL}^\star_{sc}$ equivalent formula for φ.

We have $\mathsf{GL} <_{ex} \mathsf{ATL}^\star_{sc}$ because the ATL_{sc} formula $\langle\!\langle A_1\rangle\!\rangle \mathbf{X}\,(\langle\!\langle A_2\rangle\!\rangle\ \mathbf{X}\, b \wedge \langle\!\langle A_3\rangle\!\rangle \mathbf{X}\, a)$ has no equivalent in GL. Indeed consider the CGSs $\mathcal{S}_1$ and $\mathcal{S}_2$ in Figure 2. They satisfy the same GL formulas, since move 3 for Player 1 (in $\mathcal{S}_2$) does not affect the sets of execution trees induced by all strategies for a fixed coalition: for any coalition A and state q, we have $\mathsf{ExecTree}(q, \mathsf{Strat}_{\mathcal{S}_1}(A)) = \mathsf{ExecTree}(q, \mathsf{Strat}_{\mathcal{S}_2}(A))$. Yet this move ensures that s'_0 satisfies $\langle\!\langle A_1\rangle\!\rangle \mathbf{X}\,(\langle\!\langle A_2\rangle\!\rangle \mathbf{X}\, b \wedge \langle\!\langle A_3\rangle\!\rangle \mathbf{X}\, a)$ (when players 2 and 3 respectively choose moves 2 and 1), while s_0 does not. □

Comparison with Strategy Logic [6]. Strategy Logic has been defined in [6] as an extension of LTL with first-order quantification on strategies. That player A has a strategy to enforce φ is then written $\exists\sigma_A.\ \forall\sigma_B.\ \varphi(\sigma_A, \sigma_B)$ where the arguments (*i.e.*, the strategies for the two players) given to φ indicate on which paths φ is evaluated.

While this logic has only been defined on 2-player turn-based games, its definition can easily be extended to our n-player CGS framework. We conjecture that $\mathsf{ATL}_{sc,\infty}$ and SL are incomparable (proving those results seems to be especially challenging due to the particular syntax of SL):

– SL can explicitly manipulate strategies as first-order elements. It can for instance state properties such as $\exists x_1.\ \exists y_1.\ \exists x_2.\ \exists y_2.\ \big[\varphi_1(x_1, y_1) \wedge \varphi_2(x_2, y_1) \wedge \varphi_3(x_1, y_2) \wedge \varphi_4(x_2, y_2)\big]$ which (we conjecture) $\mathsf{ATL}_{sc,\infty}$ cannot express due to the circular constraint.

– on the other hand, SL requires subformulas embedded in modalities to be closed. As a consequence, formula $\exists x_1.\ \forall y_1.\ [\mathbf{G}\,(\exists y_2.\ [\mathbf{F}\,p](x_1,y_2))](x_1,y_1)$ is not an SL formula (because $\exists y_2.\ [\mathbf{F}\,p](x_1,y_2)$ is not closed), while it is expressed in $\mathsf{ATL}_{sc,\infty}$ as $\langle\!\langle A\rangle\!\rangle\,\mathbf{G}\,(\,\langle\!\langle B\rangle\!\rangle\,\mathbf{F}\,p)$.

However, it should be noticed that the simple one-alternation fragment of SL can be translated into $\mathsf{ATL}^{\star}_{sc,\infty}$. Indeed this fragment is built from formulas of the form $\exists x_1.\ \exists y_1.\ \forall x_2.\ \forall y_2.\ \big[\varphi_1(x_1,y_2)\wedge\varphi_2(x_2,y_1)\wedge\varphi_3(x_1,y_1)\big]$ [6] which we can express as $\langle\!\langle A_1\rangle\!\rangle\,\big[\varphi_1\wedge\langle\!\langle A_2\rangle\!\rangle\,(\varphi_3\wedge\rangle\!\rangle A_1\langle\!\langle\,\varphi_2)\big]$.

4 $\mathsf{ATL}_{sc,\infty}$ and $\mathsf{ATL}^{\star}_{sc,\infty}$ Model-Checking

We begin with proving that model-checking is decidable for our logic. Still, as is the case for Strategy Logic, the resulting algorithm is non-elementary. We thus mainly focus on simpler cases (namely, memoryless and bounded-memory strategies), where more efficient algorithms can be obtained.

Theorem 6. *Model checking $\mathsf{ATL}^{\star}_{sc,\infty}$ formulas over CGS is decidable.*

Proof (sketch). Our logic $\mathsf{ATL}^{\star}_{sc}$ can be translated into $\mathrm{Q}D_{\mu}$ (see [4] for more details). This yields decidability of $\mathsf{ATL}^{\star}_{sc}$. Moreover, as we will see in Section 4.2, it is possible to encode the bounded-memory strategies as memoryless strategies over an extended CGS. Since memorylessness can be expressed with $\mathrm{Q}D_{\mu}$, this provides an indirect algorithm for $\mathsf{ATL}^{\star}_{sc,\infty}$ model checking. □

4.1 Model-Checking $\mathsf{ATL}^{\star}_{sc,0}$ and $\mathsf{ATL}_{sc,0}$

Theorem 7. *The model checking problems for $\mathsf{ATL}^{\star}_{sc,0}$ and $\mathsf{ATL}_{sc,0}$ over CGSs are* PSPACE*-complete.*

Proof. We only address the membership in PSPACE. The hardness proof is similar to that of [3] (and is also detailled in [4]).

Let $\mathcal{C}$ be a CGS, ℓ a location and F a memoryless strategy context, assigning a memoryless strategy to each player of some coalition A. Since F contains only memoryless strategies, it associates with each location one move for each agent in A. Dropping the other moves of those agents, we get a CGS, denoted $(\mathcal{C},F)$, whose set of executions is exactly the set of outcomes of F in $\mathcal{C}$.

From this and the fact that a memoryless strategy can be stored in space $O(|Q|)$, we get a simple PSPACE model-checking algorithm for $\mathsf{ATL}^{\star}_{sc,0}$ that relies on a (PSPACE) model-checking algorithm for LTL. The main difficulty is that strategy contexts prevent us from proceeding in a standard bottom-up fashion. As a consequence, our algorithm consists in enumerating strategies starting from outermost strategy quantifiers.

If φ is an $\mathsf{ATL}^{\star}_{sc,0}$ path formula, we denote by $\Phi(\varphi)$ the set of outermost *quantified* φ subformulae (*i.e.* of the form $\langle\!\langle A\rangle\!\rangle\,\psi$), and by $\sigma(\varphi)$ the corresponding LTL formula where all subformulae $\psi\in\Phi(\varphi)$ have been replaced by new

propositions a_ψ. We enumerate all possible contexts, recursively calling the algorithm at each step of the enumeration, and thus gradually taking care of each labelling a_ψ. Algorithm 1 describes the procedure. □

Algorithm 1. : $\mathsf{MC\text{-}ATL}^\star_{sc,0}(\mathcal{C}, F, \ell_0, \varphi)$ – $\mathsf{ATL}^\star_{sc,0}$ model checking

Require: a *CGS* $\mathcal{C}$, $F \in \mathsf{Strat}^0(A)$, $l_0 \in \mathsf{Loc}$ and an $\mathsf{ATL}^\star_{sc,0}$ path formula φ
Ensure: YES iff $\forall \lambda \in \mathsf{Out}(\ell_0, F), \lambda \models_F \varphi$
  $\mathcal{C}' := (\mathcal{C}, F)$
  foreach $\psi \in \Phi(\varphi)$ **do**
    case $\psi = \langle\cdot B^0 \cdot\rangle\, \psi'$:
      for $F_B \in \mathsf{Strat}^0(B)$, $\ell \in \mathsf{Loc}$ **do**
        if $\mathsf{MC\text{-}ATL}^\star_{sc,0}(\mathcal{C}, F_B \circ F, l, \psi')$, **then** label l with a_ψ
    case $\psi = \rangle B \langle\, \psi'$:
      for $l \in \mathsf{Loc}$ **do**
        if $\mathsf{MC\text{-}ATL}^\star_{sc,0}(\mathcal{C}, F_{\setminus B}, l, \psi')$, **then** label l with a_ψ
  return MC LTL $(\mathcal{C}', l_0, \mathbf{A}\sigma(\varphi))$

Remark 1. Note that PSPACE-completeness straightforwardly extends to "memoryless" extensions (*i.e.*, with quantification over memoryless strategies) of $\mathsf{ATL}^\star$ and SL. Since ATL objectives do not require memory, ATL_0 is the same as ATL, and its model-checking problem is PTIME-complete. Moreover a similar algorithm would work for *symbolic CGSs*, a succinct encoding of CGS proposed in [8,10]. Also notice that both the above algorithm and the PSPACE-hardness proof can be adapted to IATL. This corrects the Δ_2^{P}-completeness result of [1].

4.2 Bounded-Memory Strategies

The case of bounded-memory strategies can be handled in a similar way as memoryless strategies. Indeed as explained in Section 2.2, we can see an s-bounded strategy for Player A_i as a memoryless strategy over an extended structure containing the original CGS $\mathcal{C}$ and a particular CGS controlled by A_i and describing its memory. Formally, for a player A_i, we define the CGS $\mathbb{M}^s_{A_i}$ as follows: $\mathbb{M}^s_{A_i} = (\mathsf{Agt}, \mathsf{Loc}^i_s, \varnothing, \mathsf{Loc}^i_s \times \mathsf{Loc}^i_s, \varnothing, \mathcal{M}^i_s \cup \{\bot\}, \mathsf{Mov}^i_s, \mathsf{Edg}^i_s)$ where

- $\mathsf{Loc}^i_s = \{0, \ldots, s\}$ is the set of (unlabeled) locations;
- $\mathcal{M}^i_s$ is isomorphic to Loc^i_s (and we identify both sets),
- Mov^i_s and Edg^i_s do not depend on the location: Mov^i_s allows only one move $\bot$ to each player, except for player A_i, who is allowed to play any move in $\mathcal{M}^i_s$. Then Edg^i_s returns the location chosen by A_i.

Let $\overline{s} \in \mathbb{N}^{\mathsf{Agt}}$ be a memory-bound vector. Now considering the product structure $\mathcal{C}_{\overline{s}} = \prod_{A_i \in \mathsf{Agt}} \mathbb{M}^{\overline{s}(A_i)}_{A_i} \times \mathcal{C}$, for all players A_j we can very simply export $\overline{s}(A_j)$-memory-bounded strategies of $\mathcal{C}$ to *some* memoryless strategies over $\mathcal{C}_{\overline{s}}$. Indeed, given a player A_j, we do not want to consider all memoryless strategies f over $\mathcal{C}_{\overline{s}}$ but only the ones where A_j exclusively uses the information from $\mathbb{M}^{\overline{s}(A_j)}_{A_j}$

(i.e., such that $f(i_1, \ldots, i_j, \ldots, i_k, l) = f(0, \ldots, i_j, \ldots, 0, l)$). Let $\mathsf{RStrat}^0_{\mathcal{C}_{\overline{s}}}(A_j)$ be this restricted set of such strategies ; clearly we have $\mathsf{RStrat}^0_{\mathcal{C}_{\overline{s}}}(A_j) \subseteq \mathsf{Strat}^0_{\mathcal{C}_{\overline{s}}}(A_j)$. Adapting the proof of Theorem 7 to memory-bounded strategies, we get:

Proposition 8. *Let $C = (\mathsf{Agt}, \mathsf{Loc}, \ell_0, \mathsf{AP}, \mathsf{Lab}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edg})$ be a CGS. Let $\varphi \in \mathsf{ATL}^\star_{sc,b}$ involving only $\overline{s}$-memory quantifiers. Then φ can be checked in exponential space.*

Proof. We run the algorithm of Theorem 7 over the structure $\mathcal{C}_{\overline{s}}$, restricting the enumerations of $\mathsf{Strat}^0_{\mathcal{C}_{\overline{s}}}(B)$ to those of $\mathsf{RStrat}^0_{\mathcal{C}_{\overline{s}}}(B)$. □

Remark 2.
- If the memory-bounds $\overline{s}$ were given in unary, our algorithm would be PSPACE, since the LTL model-checking over the product structure can be performed on-the-fly.
- Note that this algorithm can deal with formulas containing several subformulas $\langle\!\langle A, \overline{s_1} \rangle\!\rangle \varphi_1, \ldots, \langle\!\langle A, \overline{s_p} \rangle\!\rangle \varphi_p$ with different memory bounds s_i (for the same coalition A).
- Since our algorithm consists in enumerating the strategies, it could cope with games of incomplete information, where the strategies would be based on (some of) the atomic propositions labeling a location, rather than on the location itself [15].
- Bounded-memory quantification can be defined also for the other formalisms where memory-based strategies are needed, *e.g.* $\mathsf{ATL}^\star$ or SL. Our EXPSPACE algorithm could easily be adapted to that case.

5 Conclusion

In this paper we propose powerful extensions of ATL and $\mathsf{ATL}^\star$ logics. These extensions allow us to express many interesting and complex properties that have motivated the definition of new formalisms in the past. An advantage of these extensions is to treat strategies through modalities as in ATL and $\mathsf{ATL}^\star$.

As future work, we plan to study the exact complexity of model-checking $\mathsf{ATL}_{sc,\infty}$ and $\mathsf{ATL}^\star_{sc,\infty}$, with the aim of finding reasonably efficient algorithms for fragments of these expressive logics. Finally we think that the ability to deal explicitly with bounded-memory strategies is an interesting approach to develop.

References

1. Ågotnes, T., Goranko, V., Jamroga, W.: Alternating-time temporal logics with irrevocable strategies. In: Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK 2007), pp. 15–24 (June 2007)
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)
3. Baier, C., Brázdil, T., Größer, M., Kučera, A.: Stochastic game logic. In: Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST 2007), pp. 227–236. IEEE Comp. Soc. Press, Los Alamitos (2007)

4. Brihaye, T., Costa, A.D., Laroussinie, F., Markey, N.: ATL with strategy contexts and bounded memory. Technical Report LSV-08-14, Lab. Specification et Verification (February 2008)
5. Chatterjee, K., Henzinger, T.A., Jurdziński, M.: Games with secure equilibria. Theoretical Computer Science 365(1-2), 67–82 (2006)
6. Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 59–73. Springer, Heidelberg (2007)
7. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronous skeletons using branching-time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
8. Jamroga, W., Dix, J.: Do agents make model checking explode (computationally)? In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS, vol. 3690, pp. 398–407. Springer, Heidelberg (2005)
9. Kupferman, O., Vardi, M.Y., Wolper, P.: Module checking. Information and Computation 164(2), 322–344 (2001)
10. Laroussinie, F., Markey, N., Oreiby, G.: On the expressiveness and complexity of ATL. In: Seidl, H. (ed.) FOSSACS 2007. LNCS, vol. 4423, pp. 243–257. Springer, Heidelberg (2007)
11. Mazala, R.: Infinite games. In: Grädel, E., Thomas, W., Wilke, T. (eds.) Automata, Logics, and Infinite Games. LNCS, vol. 2500, pp. 23–42. Springer, Heidelberg (2002)
12. Pinchinat, S.: A generic constructive solution for concurrent games with expressive constraints on strategies. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 253–267. Springer, Heidelberg (2007)
13. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS 1977), pp. 46–57. IEEE Comp. Soc. Press, Los Alamitos (1977)
14. Queille, J.-P., Sifakis, J.: Specification and verification of concurrent systems in CESAR. In: Dezani-Ciancaglini, M., Montanari, U. (eds.) Programming 1982. LNCS, vol. 137, pp. 337–351. Springer, Heidelberg (1982)
15. Schobbens, P.-Y.: Alternating-time logic with imperfect recall. In: Proceedings of the 1st Workshop on Logic and Communication in Multi-Agent Systems (LCMAS 2003). ENTCS, vol. 85. Elsevier, Amsterdam (2004)
16. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)

A Relational Model of a Parallel and Non-deterministic λ-Calculus

Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto

Laboratoire PPS, Université Paris 7,
2, place Jussieu (case 7014), 75251 Paris Cedex 05, France
{antonio.bucciarelli,thomas.ehrhard,giulio.manzonetto}@pps.jussieu.fr

Abstract. We recently introduced an extensional model of the pure λ-calculus living in a canonical cartesian closed category of sets and relations [6]. In the present paper, we study the non-deterministic features of this model. Unlike most traditional approaches, our way of interpreting non-determinism does not require any additional powerdomain construction. We show that our model provides a straightforward semantics of *non-determinism* (*may* convergence) by means of *unions* of interpretations, as well as of *parallelism* (*must* convergence) by means of a binary, non-idempotent operation available on the model, which is related to the *mix rule* of Linear Logic. More precisely, we introduce a λ-calculus extended with non-deterministic choice and parallel composition, and we define its operational semantics (based on the *may* and *must* intuitions underlying our two additional operations). We describe the interpretation of this calculus in our model and show that this interpretation is sensible with respect to our operational semantics: a term converges if, and only if, it has a non-empty interpretation.

Keywords: λ-calculus, relational model, non-determinism, parallel composition, denotational semantics.

1 Introduction

Pure and typed λ-terms are specifications of sequential and deterministic processes. Several extensions of the λ-calculus with parallel and/or non-deterministic constructs have been proposed in the literature, either to increase the expressive power of the language, in the typed [19,17,14] and untyped [4,5] settings, or to study the interplay between higher order features and parallel/non-deterministic features [16,8,9].

When introducing non-determinism in a functional setting, it is crucial to specify what notion of convergence is chosen. Two widely used notions are:

- the *must* convergence: a non-deterministic choice converges if all its components do. This characterizes the *demonic* non-determinism.
- the *may* convergence: a non-deterministic choice converges if at least one of its components does. This characterizes the *angelic* non-determinism.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 107–121, 2009.

The usual denotational models of functional calculi do not accommodate *may* non-determinism: let TRUE and FALSE be two convergent terms[1], whose denotations in standard models are distinct. What semantic value should take the non-deterministic term TRUE + FALSE, which *may* converges to TRUE and to FALSE? The value should be both TRUE and FALSE if we want the semantics to be invariant under reduction!

The typical way of interpreting "multi-valued" terms, like the one above, is to use models based on *powerdomains* [18], often defined as filter models with respect to suitable notions of intersection and union types [8,9]. The semantics of TRUE + FALSE becomes some kind of join of both values, available in the powerdomain (similar techniques are also used for interpreting *must* non-determinism). In this framework, both kinds of non-determinism are modelled by some idempotent, commutative and associative operations.

In a recent paper [11], Faure and Miquel define a categorical counterpart of the syntactical notion of parallel execution: the *aggregation monad*. Powerdomains, sets with union and multisets with multi-union are all instances of aggregation monads (in categories of domains and of sets, respectively). In general, the notion of parallel composition modelled by an aggregation monad is neither idempotent, nor commutative, nor associative.

There are however models of the ordinary λ-calculus where aggregation, considered as parallel composition (that is, as *must* non-determinism), can be interpreted without introducing any additional structure, such as the above mentioned aggregation monads or powerdomain constructions.

This is the case in models of multiplicative exponential linear logic (MELL), where aggregation can be interpreted by the *mix rule*, if available. This rule allows to "put together" any two proofs whatsoever [7]. More precisely, parallel composition is obtained by combining the mix rule with the contraction rule. Indeed, mix can be seen as a linear morphism $X \otimes Y \multimap X \parr Y$, so that there is a morphism $?A \otimes ?A \multimap ?A$, obtained by composing the mix morphism $?A \otimes ?A \multimap ?A \parr ?A$ with the contraction morphism $?A \parr ?A \multimap ?A$. This composite morphism defines a commutative algebra structure on $?A$, which is used to model the "parallel composition" of MELL proofs. Thus, to obtain a model of parallel λ-calculus, it is sufficient to solve the equation $\mathcal{D} \cong \mathcal{D} \Rightarrow \mathcal{D}$, with an object $\mathcal{D}$ of shape $?A$.

This is precisely what we did in [6], in a particularly simple model of linear logic: the model of sets and relations. Similar constructions are possible in other, richer models, such as the well known model of coherence spaces [12], or the model of hypercoherences [10]: the mix rule is available there, as well as in many other models. This shows that coherence (which prevents the above join of TRUE and FALSE) is not an obstacle to the interpretation of the *must* non-determinism in the pure λ-calculus[2]. Our model $\mathcal{D}$ of [6] satisfies the recursive equation $\mathcal{D} = ?(A)$ where $A = (\mathcal{D}^{\mathbb{N}})^{\perp}$, and therefore, $\mathcal{D}$ has the commutative

[1] They could be the actual boolean constants in a typed λ-calculus with constants, or the projections $\lambda xy.x$, $\lambda xy.y$ as pure λ-terms.

[2] In a typed language like PCF, this would be more problematic, since the object interpreting the type of booleans does not have the above mentioned structure.

algebra structure mentioned above. It is precisely this structure that we use for interpreting parallel composition, just as Danos and Krivine did in [7] for an extension of $\lambda\mu$-calculus with a parallel composition operation.

However, the category of sets and relations has another feature, which allows for a direct interpretation of the *may* non-determinism as well: morphisms are arbitrary relations between sets (interpreting types), and hence morphisms are closed under arbitrary unions. Thanks to this union operation on morphisms, *may* non-determinism can be interpreted directly, without introducing any additional powerdomain construction or aggregation monad. Of course, this operation is not available in the coherence or hypercoherence space models. Note that, if we consider $M + N \to M$ as a reduction rule of our calculus, then our semantics is not invariant under reduction, since the process of performing non-deterministic choices entails a non recoverable loss of information. But the situation is fundamentally similar with the powerdomain-based interpretations.

To summarize, in our model $\mathcal{D}$, the semantic counterparts of *may* and *must* non-determinism are at hand: they are simply the set-theoretic union and the mix-based algebraic operation. In this framework, parallel composition is no longer idempotent. This is quite natural if we consider each component of a parallel composition as the specification of a process whose execution requires the consumption of some kind of resources.

Contents. We introduce an extension of λ-calculus with parallel composition and non-deterministic choice, called $\lambda_{+\|}$*-calculus*, and we define its operational semantics by associating with each term a generalized hnf (head normal form), which is a set of multisets of terms whose head subterms are variables[3]. Roughly speaking, the operational value of a term is the collection of all possible outcomes of its head reductions. When the head subterm is $M+N$ (*may* non-deterministic choice), the head reduction goes on by choosing either M or N, and when the head subterm is $M\|N$ (*must* parallelism), the head reduction forks.

We provide the denotational semantics of the $\lambda_{+\|}$-calculus in $\mathcal{D}$, considered as a λ-model, and endowed with two additional operations which turn it into a semiring. We prove the soundness with respect to β-reduction, and we show that the interpretations of the hnf's of a term M are included in the interpretation of M. Next, we generalize Krivine's realizability technique to our extended calculus, showing that our denotational model is *sensible*: the operational value of a term is non-empty (i.e., a term is solvable) if, and only if, its denotation is non-empty.

2 Preliminaries

To keep this article self-contained we summarize some definitions and results that will be used in the sequel. In particular, we present our semantic framework **MRel** and we recall the construction of a specific reflexive object $\mathcal{D}$ of **MRel**, which we have introduced in [6]. Our main reference for category theory is [1].

[3] This is reminiscent of the *capability semantics* of [8], but we consider different notions of convergence and of head normal form.

2.1 Multisets and Sequences

Let S be a set. We denote by $\mathcal{P}(S)$ the collection of all subsets of S. A *multiset* m over S can be defined as an unordered list $m = [a_1, a_2, \ldots]$ with repetitions such that $a_i \in S$ for all i. A multiset m is called *finite* if it is a finite list, we denote by $[]$ the empty multiset. Given two multisets $m_1 = [a_1, a_2, \ldots]$ and $m_2 = [b_1, b_2, \ldots]$ the *multi-union* of m_1, m_2 is defined by $m_1 \uplus m_2 = [a_1, b_1, a_2, b_2, \ldots]$. We will write $\mathcal{M}_f(S)$ for the set of all finite multisets over S.

We denote by $\mathbb{N}$ the set of natural numbers. Given two $\mathbb{N}$-indexed sequences $\sigma = (\sigma_1, \sigma_2, \ldots), \tau = (\tau_1, \tau_2, \ldots)$ of multisets we define the *multi-union* of σ and τ componentwise as $\sigma \bar{\uplus} \tau = (\sigma_1 \uplus \tau_1, \sigma_2 \uplus \tau_2, \ldots)$. An $\mathbb{N}$-indexed sequence $\sigma = (m_1, m_2, \ldots)$ of multisets is *quasi-finite* if $m_i = []$ holds for all, but a finite number of indices i. If S is a set, then we denote by $\mathcal{M}_f(S)^{(\omega)}$ the set of all quasi-finite $\mathbb{N}$-indexed sequences of multisets over S. We write $\star$ for the $\mathbb{N}$-indexed sequence of empty multisets, i.e., $\star$ is the only inhabitant of $\mathcal{M}_f(\emptyset)^{(\omega)}$.

2.2 MRel: A Cartesian Closed Category of Sets and Relations

We now present the category **MRel**, which is the Kleisli category of the functor $\mathcal{M}_f(-)$ over the $\star$-autonomous category **Rel** of sets and relations. We provide here a direct definition, since in the sequel we will not use explicitly the monoidal structure of **Rel**.

- The objects of **MRel** are all the sets.
- A morphism from S to T is a relation from $\mathcal{M}_f(S)$ to T, in other words, $\mathbf{MRel}(S, T) = \mathcal{P}(\mathcal{M}_f(S) \times T)$.
- The identity of S is the relation $Id_S = \{([a], a) \mid a \in S\} \in \mathbf{MRel}(S, S)$.
- The composition of $s \in \mathbf{MRel}(S, T)$ and $t \in \mathbf{MRel}(T, U)$ is defined by:

$$t \circ s = \{(m, c) \mid \exists (m_1, b_1), \ldots, (m_k, b_k) \in s \text{ such that } m = m_1 \uplus \ldots \uplus m_k \text{ and } ([b_1, \ldots, b_k], c) \in t\}.$$

We now provide an overview of the proof of cartesian closedness.

Theorem 1. *The category* **MRel** *is cartesian closed.*

Proof. The terminal object $\mathbb{1}$ is the empty set $\emptyset$, and the unique element of $\mathbf{MRel}(S, \emptyset)$ is the empty relation.

Given two sets S_1 and S_2, their categorical product $S_1 \& S_2$ in **MRel** is their disjoint union:

$$S_1 \& S_2 = (\{1\} \times S_1) \cup (\{2\} \times S_2)$$

and the projections π_1, π_2 are given by:

$$\pi_i = \{([(i, a)], a) \mid a \in S_i\} \in \mathbf{MRel}(S_1 \& S_2, S_i), \text{ for } i = 1, 2.$$

Given $s \in \mathbf{MRel}(U, S_1)$ and $t \in \mathbf{MRel}(U, S_2)$, the corresponding morphism $\langle s, t \rangle \in \mathbf{MRel}(U, S_1 \& S_2)$ is given by:

$$\langle s, t \rangle = \{(m, (1, a)) \mid (m, a) \in s\} \cup \{(m, (2, b)) \mid (m, b) \in t\}.$$

We will consider the canonical bijection between $\mathcal{M}_f(S_1) \times \mathcal{M}_f(S_2)$ and $\mathcal{M}_f(S_1 \,\&\, S_2)$ as an equality, hence we will still denote by (m_1, m_2) the corresponding element of $\mathcal{M}_f(S_1 \& S_2)$.

Given two objects S and T the exponential object $S \Rightarrow T$ is $\mathcal{M}_f(S) \times T$ and the evaluation morphism is given by:

$$eval_{ST} = \{(([(m,b)],m),b) \mid m \in \mathcal{M}_f(S) \text{ and } b \in T\} \in \mathbf{MRel}((S \Rightarrow T) \& S, T)\,.$$

Given any set U and any morphism $s \in \mathbf{MRel}(U \& S, T)$, there is exactly one morphism $\Lambda(s) \in \mathbf{MRel}(U, S \Rightarrow T)$ such that:

$$eval_{ST} \circ \langle \Lambda(s), Id_S \rangle = s,$$

namely, $\Lambda(s) = \{(p,(m,b)) \mid ((p,m),b) \in s\}$. □

The points of an object S, i.e., the elements of $\mathbf{MRel}(\mathbb{1}, S)$, are relations between $\mathcal{M}_f(\emptyset)$ and S. These are, up to isomorphism, the subsets of S.

2.3 An Extensional Reflexive Object in MRel

A *reflexive object* of a cartesian closed category $\mathbf{C}$ (ccc, for short) is a triple $\mathcal{U} = (U, \mathcal{A}, \lambda)$ such that U is an object of $\mathbf{C}$, and $\lambda \in \mathbf{C}(U \Rightarrow U, U)$ and $\mathcal{A} \in \mathbf{C}(U, U \Rightarrow U)$ satisfy $\mathcal{A} \circ \lambda = Id_{U \Rightarrow U}$. $\mathcal{U}$ is called *extensional* if, moreover, $\lambda \circ \mathcal{A} = Id_U$; in this case we have that $U \cong U \Rightarrow U$.

We define a reflexive object $\mathcal{D}$ in **MRel**, which is extensional by construction. We let $(D_n)_{n \in \mathbb{N}}$ be the increasing family of sets defined by:

- $D_0 = \emptyset$,
- $D_{n+1} = \mathcal{M}_f(D_n)^{(\omega)}$.

Finally, we set $D = \bigcup_{n \in \mathbb{N}} D_n$. So we have $D_0 = \emptyset$ and $D_1 = \{\star\} = \{([], [], \dots)\}$. The elements of D_2 are quasi-finite sequences of multisets over a singleton, i.e., quasi-finite sequences of natural numbers, and so on.

We say that $\sigma \in D$ *has rank* n if $n \in \mathbb{N}$ is minimum such that $\sigma \in D_n$.

In order to define an isomorphism in **MRel** between D and $D \Rightarrow D = \mathcal{M}_f(D) \times D$ just notice that every element $\sigma = (\sigma_1, \sigma_2, \dots) \in D$ stands for the pair $(\sigma_1, (\sigma_2, \dots))$ and *vice versa*.Given $\sigma \in D$ and $m \in \mathcal{M}_f(D)$, we write $m :: \sigma$ for the element $\tau = (\tau_1, \tau_2, \dots) \in D$ such that $\tau_1 = m$ and $\tau_{i+1} = \sigma_i$. This defines a bijection between $\mathcal{M}_f(D) \times D$ and D, and hence an isomorphism in **MRel** as follows:

Proposition 1. *(Bucciarelli, et al. [6]) The triple $\mathcal{D} = (D, \mathcal{A}, \lambda)$ where:*

- $\lambda = \{([(m,\sigma)], m :: \sigma) \mid m \in \mathcal{M}_f(D), \sigma \in D\} \in \mathbf{MRel}(D \Rightarrow D, D)$,
- $\mathcal{A} = \{([m :: \sigma], (m,\sigma)) \mid m \in \mathcal{M}_f(D), \sigma \in D\} \in \mathbf{MRel}(D, D \Rightarrow D)$,

is an extensional reflexive object of **MRel**.

3 A Parallel and Non-deterministic λ-Calculus

In this section we introduce the syntax and the operational semantics of a parallel and non-deterministic extension of λ-calculus that we call $\lambda_{+\|}$*-calculus*.

3.1 Syntax of $\lambda_{+\|}$-Calculus

To begin with, we define the set $\Lambda_{+\|}$ of λ-terms enriched with two binary operators $+$ and $\|$, that is the set of terms generated by the following grammar (where x ranges over a countable set Var of variables):

$$M, N \ ::= \ x \mid \lambda x.M \mid MN \mid M + N \mid M \| N \ .$$

The elements of $\Lambda_{+\|}$ are called $\lambda_{+\|}$-*terms* and will be denoted by $M, N, P, \ldots$ Intuitively, $M+N$ denotes the *non-deterministic choice* between M and N, and $M\|N$ stands for their *parallel composition*.

As usual, we suppose that application associates to the left and λ-abstraction to the right. Moreover, to lighten the notation, we assume that application and λ-abstraction take precedence over $+$ and $\|$. The notions of *free* and *bound* variables of a term are defined in the obvious way.

A *substitution* is a finite set $s = \{(x_1, N_1), \ldots, (x_k, N_k)\}$ such that $x_i \neq x_j$ for all $1 \leq i < j \leq k$. Given a $\lambda_{+\|}$-term M and a substitution s as above, we denote by Ms the term obtained by substituting *simultaneously* the term N_j for all free occurrences of x_j (for $1 \leq j \leq k$) in M, subject to the usual proviso about renaming bound variables in M to avoid capture of free variables in the N_j's. If $s = \{(x, N)\}$ we will write $M[N/x]$ for Ms.

Note that, in general, $M\{(x_1, N_1), \ldots, (x_k, N_k)\} \neq M[N_1/x_1] \cdots [N_k/x_k]$. For instance, $x\{(x, y), (y, z)\} = y$, whereas $x[y/x][z/y] = z$. Actually, k-ary substitutions will be only used in Section 5 in the proof of the adequacy lemma.

As a matter of notation, we will write $\vec{P}$ for a (possibly empty) finite sequence of $\lambda_{+\|}$-terms $P_1 \ldots P_k$ and $\ell(\vec{P})$ for the length of $\vec{P}$. It is easy to check that every $\lambda_{+\|}$-term M has the form $\lambda\vec{x}.N\vec{P}$ where N, which is called *the head subterm of* M, is either a variable, a non-deterministic choice, a parallel composition or a λ-abstraction. Notice that, in this last case, we must have $\ell(\vec{P}) > 0$.

3.2 Operational Semantics

The set $\Lambda^h_{+\|} \subset \Lambda_{+\|}$ of *head normal forms*[4] (*hnf*'s, for short) is the set of $\lambda_{+\|}$-terms whose head subterm is a variable (called *head variable*).

The intuitive idea of the head reduction of $\lambda_{+\|}$-calculus underlying the notion of "value" (formalized below) is the following:

- when a term has the head subterm of the form $N_1 + N_2$, either of the alternatives may be chosen to pursue the head reduction, and the final value is the union of the values obtained by each choice. In particular, if one of the choices produces a non-empty value, then the global value is non-empty.
- when a term has the head subterm of the form $N_1\|N_2$, the head reduction forks, and the final value is obtained by "mixing" the values eventually obtained. In particular, if the value of one of the subprocesses is empty, then also the global value is.

[4] This terminology is coherent with the one usually adopted for λ-calculus (see [2, Def. 2.2.11]).

Instead of defining the operational semantics of $\lambda_{+\|}$-calculus via an explicit (head) rewriting system, we associate with each $M \in \Lambda_{+\|}$ the value eventually obtained by head reducing M. In particular, we use union (resp. multi-union) to get the value of $M_1 + M_2$ (resp. $M_1 \| M_2$) out of the values of M_1 and M_2.

Definition 1. *A* multiple hnf *is a finite multiset of hnf's of $\lambda_{+\|}$-calculus. A* value *is a set of multiple hnf's.*

To help the reader to get familiar with these notions, we first provide some simple examples of values (where[5] $\mathbf{I} \equiv \lambda x.x$, $\Delta \equiv \lambda x.xx$ and $\Omega \equiv \Delta\Delta$):

- the value of $\mathbf{I} + \Delta$ is $\{[\mathbf{I}], [\Delta]\}$. In other words, the term $\mathbf{I} + \Delta$ has two different multiple hnf's, which are singleton multisets;
- the value of $\mathbf{I} \| \Delta$ is $\{[\mathbf{I}, \Delta]\}$, then $\mathbf{I} \| \Delta$ has just one multiple hnf;
- the values of $\mathbf{I} + \Omega$ and $\mathbf{I} \| \Omega$ are $\{[\mathbf{I}]\}$ and $\emptyset$, respectively. This is a consequence of the fact that the value of Ω is the empty-set.

In general, the value $H(M)$ of a $\lambda_{+\|}$-term M can be characterized as the limit of an increasing sequence $(H_n(M))_{n \in \mathbb{N}}$ of "partial" values, which are defined by induction on $n \in \mathbb{N}$ and by cases on the form of the head subterm of M.

Definition 2. *Let $M \equiv \lambda \vec{x}.N\vec{P}$ be a $\lambda_{+\|}$-term.*

- $H_0(M) = \emptyset$;
- $H_{n+1}(M) = \begin{cases} \{[M]\} & \text{if } N \equiv y, \\ H_n(\lambda \vec{x}.Q[P_1/y]P_2 \cdots P_{\ell(\vec{P})}) & \text{if } N \equiv \lambda y.Q, \\ H_n(\lambda \vec{x}.N_1\vec{P}) \cup H_n(\lambda \vec{x}.N_2\vec{P}) & \text{if } N \equiv N_1 + N_2, \\ \{m_1 \uplus m_2 \mid \exists m_i \in H_n(\lambda \vec{x}.N_i\vec{P}) \text{ for } i = 1, 2\} & \text{if } N \equiv N_1 \| N_2. \end{cases}$

Notice that, for all $M \in \Lambda_{+\|}$ and $n \in \mathbb{N}$, the value $H_n(M) \subset \mathcal{M}_f(\Lambda^h_{+\|})$ is a finite set of multiple hnf's. Since the sequence $(H_n(M))_{n \in \mathbb{N}}$ is increasing, we can define the (final) value of M as its limit.

Definition 3. *The* value of a $\lambda_{+\|}$-term M *is defined by* $H(M) = \bigcup_{n \in \mathbb{N}} H_n(M)$.

Of course, $H(M)$ may be infinite as shown in the example below.

Example 1. Consider the $\lambda_{+\|}$-term $M \equiv \lambda n.\underline{0} + \mathbf{s}n$, where $\underline{0} \equiv \lambda xy.y$ is the 0-th Church numeral and $\mathbf{s} \equiv \lambda nxy.nx(xy)$ implements the successor function. Let now $C \equiv \mathbf{Y}M$ where $\mathbf{Y}$ is some fixpoint combinator. To have simpler calculations, we suppose that $\mathbf{Y}M$ reduces to $M(\mathbf{Y}M)$ in just one step of head β-reduction. Then, we get:

- $H_0(C) = \emptyset$,
- $H_1(C) = H_0(MC) = \emptyset$,
- $H_2(C) = H_1(MC) = H_0(\underline{0} + \mathbf{s}C) = \emptyset$,
- $H_3(C) = H_2(MC) = H_1(\underline{0} + \mathbf{s}C) = \{[\underline{0}]\} \cup H_0(\mathbf{s}C) = \{[\underline{0}]\}$.

Pursuing the calculation a little further, one gets $H_9(C) = \{[\underline{0}], [\underline{1}]\}$ and, eventually, $H(C) = \{[\underline{n}] \mid n \in \mathbb{N}\}$.

[5] The symbol $\equiv$ denotes syntactical equality.

3.3 Solvability

We now present the natural notion of solvability for the $\lambda_{+\|}$-calculus.

Definition 4. *A $\lambda_{+\|}$-term M is* solvable *if $H(M) \neq \emptyset$. The set of solvable terms will be denoted by $\mathcal{N}$.*

Among solvable terms, we single out the set $\mathcal{N}_0$ of hnf's starting with a variable, and the set $\mathcal{N}_1$ of solvable terms having a multiple hnf whose head variables are free.

Definition 5. *We set:*

- $\mathcal{N}_0 = \{x\vec{P} \mid x \in \mathrm{Var}$ *and* $\vec{P} \in \Lambda_{+\|}\}$, *and*
- $\mathcal{N}_1 = \{M \in \Lambda_{+\|} \mid \exists[\lambda\vec{x}_1.y_1\vec{P}_1, \ldots, \lambda\vec{x}_k.y_k\vec{P}_k] \in H(M) \wedge (\forall j = 1..k)\ y_j \notin \vec{x}_j\}$.

We end this section stating a technical proposition, which will be useful in Section 5. The proof is not difficult (but quite long) and it is omitted.

Proposition 2. *Let $M \in \Lambda_{+\|}$ and $x \in \mathrm{Var}$, then we have that:*

(i) if $Mx \in \mathcal{N}$ then $M \in \mathcal{N}$,
(ii) if $M\Omega \in \mathcal{N}_1$ then $M \in \mathcal{N}_1$,
(iii) if $M \in \mathcal{N}_1$ then $MN \in \mathcal{N}_1$ for all $N \in \Lambda_{+\|}$.

Notice that in the case of the pure λ-calculus the analogous properties are trivial.

4 A Relational Model of $\lambda_{+\|}$-Calculus

Exploiting the existence of countable products in **MRel** we have shown in [6] that the reflexive object $\mathcal{D} = (D, \mathcal{A}, \lambda)$ built in Section 2.3 can be turned into a λ-*model* [2, Def. 5.2.1] (this was not clear before, since the category **MRel** does not have *enough points* [1, Def. 2.1.4]). The underlying set of the λ-model associated with D by our construction is the set of "finitary" morphisms in $\mathbf{MRel}(D^{\mathrm{Var}}, D)$, where D^{Var} is the Var-indexed categorical product of countably many copies of D.

4.1 Finitary Morphisms in MRel

The morphisms in $\mathbf{MRel}(D^{\mathrm{Var}}, D)$ are sets of pairs whose first projection is a finite multiset of elements in D^{Var}, and whose second projection is an element of D. Since categorical products in **MRel** are disjoint unions, a typical such pair is of the form:

$$([(x_1, \sigma_1^1), \ldots, (x_1, \sigma_1^{n_1}), \ldots, (x_k, \sigma_k^1), \ldots, (x_k, \sigma_k^{n_k})], \sigma)$$

where $k, n_1, \ldots, n_k \in \mathbb{N}$, $x_1, \ldots, x_k \in \mathrm{Var}$ and $\sigma_1^1, \ldots, \sigma_k^{n_k}, \sigma \in D$.

Notation 1. *Given* $m \in \mathcal{M}_f(D^{\mathrm{Var}})$ *and* $x \in \mathrm{Var}$*, we set* $m_x = [\sigma \mid (x,\sigma) \in m] \in \mathcal{M}_f(D)$ *and* $m_{-x} = [(y,\sigma) \in m \mid y \neq x] \in \mathcal{M}_f(D^{\mathrm{Var}})$.

In general, given an object U of a ccc $\mathbf{C}$, we say that a morphism $f \in \mathbf{C}(U^{\mathrm{Var}}, U)$ is "finitary" if it can be decomposed as $f = f_I \circ \pi_I$ for some finite set I of variables (see [6, Sec. 3.1]). Working in **MRel** it is more convenient to take the following equivalent definition.

Definition 6. *A morphism* $r \in \mathbf{MRel}(D^{\mathrm{Var}}, D)$ *is* finitary *if there exists a finite set* I *of variables such that for all* $(m,\sigma) \in r$ *and* $x \in \mathrm{Var}$ *we have that* $m_x \neq []$ *entails* $x \in I$.

We denote by $\mathbf{MRel}_f(D^{\mathrm{Var}}, D)$ the set of all finitary morphisms.

4.2 The Model

From [6, Thm. 1] we know that $(\mathbf{MRel}_f(D^{\mathrm{Var}}, D), \bullet)$, where $\bullet$ is defined as usual by $r_1 \bullet r_2 = eval \circ \langle \mathcal{A} \circ r_1, r_2 \rangle$, can be endowed with a structure of λ-model.

In order to interpret $\lambda_{+\|}$-terms as finitary morphisms of **MRel** we are going to define on $\mathbf{MRel}(D^{\mathrm{Var}}, D)$ two binary operations of *sum* and *aggregation* for modelling non-deterministic choice and parallel composition, respectively, and to prove that $\mathbf{MRel}_f(D^{\mathrm{Var}}, D)$ is closed under these operations.

Definition 7. *Let* $r_1, r_2 \in \mathbf{MRel}(D^{\mathrm{Var}}, D)$*, then:*

- *the* sum *of* r_1 *and* r_2 *is defined by* $r_1 \oplus r_2 = r_1 \cup r_2$.
- *the* aggregation *of* r_1 *and* r_2 *is defined by* $r_1 \odot r_2 = \{(m_1 \uplus m_2, \sigma_1 \bar{\uplus} \sigma_2) \mid \exists (m_i, \sigma_i) \in r_i,\ \text{for } i = 1,2\}$.

Proposition 3. *The set* $\mathbf{MRel}_f(D^{\mathrm{Var}}, D)$ *is closed under sum and aggregation.*

Proof. Straightforward. In both cases, the union of the finite sets of variables I_1 and I_2 given by the finiteness of the arguments of the operation, is a witness of the finiteness of the result. □

Composition is right-distributive over sum and aggregation.

Proposition 4. *Let* $r, s \in \mathbf{MRel}(D^{\mathrm{Var}}, D)$ *and* $t \in \mathbf{MRel}(D^{\mathrm{Var}}, D^{\mathrm{Var}})$*, then:*

- $(r \oplus s) \circ t = (r \circ t) \oplus (s \circ t)$,
- $(r \odot s) \circ t = (r \circ t) \odot (s \circ t)$.

Proof. Straightforward. □

The units of the operations $\oplus$ and $\odot$ are $0 = \emptyset$ and $1 = \{([], \star)\}$, respectively; $(\mathbf{MRel}_f(D^{\mathrm{Var}}, D), \oplus, 0)$ and $(\mathbf{MRel}_f(D^{\mathrm{Var}}, D), \odot, 1)$ are commutative monoids. Moreover, 0 annihilates $\odot$ and aggregation distributes over sum. Summing up, the following proposition gives an overview of the algebraic properties of $\mathbf{MRel}_f(D^{\mathrm{Var}}, D)$ equipped with application, sum and aggregation.

Proposition 5.
- $(\mathbf{MRel}_f(D^{\mathrm{Var}}, D), \oplus, \odot, 0, 1)$ *is a commutative semiring.*
- $\bullet$ *is right-distributive over* $\oplus$ *and* $\odot$.
- $\oplus$ *is idempotent (whereas* $\odot$ *is not).*

Proof. Straightforward.

4.3 The Absolute Interpretation

Before going through the formal definition of the interpretation of $\lambda_{+\|}$-terms, we present a short digression on the nature of such an interpretation.

In our framework, the $\lambda_{+\|}$-terms will be interpreted as morphisms in $\mathbf{MRel}_f(D^{\mathrm{Var}}, D)$, i.e., as subsets of $\mathcal{M}_f(D^{\mathrm{Var}}) \times D$. The occurrence of a particular pair $([(x_1, \sigma_1^1), \ldots, (x_1, \sigma_1^{n_1}), \ldots, (x_k, \sigma_k^1), \ldots, (x_k, \sigma_k^{n_k})], \sigma)$ in the interpretation of a term M may be read as "in an environment ρ such that $\rho(x_i) = [\sigma_i^1, \ldots, \sigma_i^{n_i}]$ (for all $i = 1, \ldots, k$) the interpretation $[\![M]\!]_\rho$ contains σ".

Hence, here there is no need of providing explicitly an environment to the interpretation function as classically done for λ-models [2, Def. 5.2.1(ii)] because the whole information is coded inside the elements of the λ-model itself.

On the other hand, the categorical interpretation of a term M is usually defined with respect to a finite list of variables, containing the free variables of M [2, Def. 5.5.3(vii)]. Intuitively, our interpretation is defined with respect to the list of *all* variables, encompassing then all categorical interpretations.

These considerations lead us to the definition of $[\![-]\!] : \Lambda_{+\|} \to \mathbf{MRel}_f(D^{\mathrm{Var}}, D)$ below, that we call the *absolute interpretation*[6] of $\lambda_{+\|}$-terms:

- $[\![x]\!] = \pi_x$, for $x \in \mathrm{Var}$,
- $[\![M_1 M_2]\!] = \mathit{eval} \circ \langle \mathcal{A} \circ [\![M_1]\!], [\![M_2]\!] \rangle$,
- $[\![\lambda x.M]\!] = \lambda \circ \Lambda([\![M]\!] \circ \eta_x)$,
- $[\![M_1 + M_2]\!] = [\![M_1]\!] \oplus [\![M_2]\!]$,
- $[\![M_1 \| M_2]\!] = [\![M_1]\!] \odot [\![M_2]\!]$,

where $\eta_x \in \mathbf{MRel}(D^{\mathrm{Var}} \& D, D^{\mathrm{Var}})$ is defined componentwise, for $y \in \mathrm{Var}$, by:

$$\pi_y \circ \eta_x = \begin{cases} \pi_2 & \text{if } x \equiv y, \\ \pi_y \circ \pi_1 & \text{if } x \not\equiv y. \end{cases}$$

In what follows, we will use the inductive characterization of the interpretation of (some) $\lambda_{+\|}$-terms provided by the proposition below:

Proposition 6

(i) $[\![x]\!] = \{([(x, \sigma)], \sigma) \mid \sigma \in D\}$,
(ii) $[\![MN]\!] = \{(m_0 \uplus m_1 \uplus \ldots \uplus m_k, \sigma) \mid \exists k \geq 0, (m_0, [\tau_1, \ldots, \tau_k] :: \sigma) \in [\![M]\!], (m_i, \tau_i) \in [\![N]\!] \ \mathit{for}\ 1 \leq i \leq k\}$,
(iii) $[\![\lambda x.M]\!] = \{(m_{-x}, m_x :: \sigma) \mid (m, \sigma) \in [\![M]\!]\}$.

Proof. Simple calculations based on the definitions of Section 2. □

We show now the soundness of the interpretation with respect to β-conversion, which relies on the following lemma.

Lemma 1. *If* $M, N \in \Lambda_{+\|}$ *and* $x \in \mathrm{Var}$*, then* $[\![M[N/x]]\!] = [\![M]\!] \circ \eta_x \circ \langle Id, [\![N]\!] \rangle$.

[6] See [15, Sec. 2.3.2] (and cf. [20]) for more details on the relations among the absolute, algebraic and categorical interpretations, and on how the former allows to recover the others.

Proof. By structural induction on M. The cases $M \equiv M_1 + M_2$ and $M \equiv M_1 \| M_2$ are settled by using Proposition 4. For the other cases, one can use Proposition 6 and the following characterization: $\eta_x \circ \langle Id, \llbracket N \rrbracket \rangle = \{([(y,\sigma)],(y,\sigma)) \mid \sigma \in D, y \not\equiv x\} \cup \{(m,(x,\sigma)) \mid (m,\sigma) \in \llbracket N \rrbracket\} \in \mathbf{MRel}(D^{\mathrm{Var}}, D^{\mathrm{Var}})$. □

Lemma 2. *(Soundness) For all $M, N \in \Lambda_{+\|}$ and $x \in \mathrm{Var}$, we have $\llbracket (\lambda x.M)N \rrbracket = \llbracket M[N/x] \rrbracket$.*

Proof. $\llbracket (\lambda x.M)N \rrbracket = eval \circ \langle \mathcal{A} \circ \lambda \circ \Lambda(\llbracket M \rrbracket \circ \eta_x), \llbracket N \rrbracket \rangle = eval \circ \langle \Lambda(\llbracket M \rrbracket \circ \eta_x), \llbracket N \rrbracket \rangle = \llbracket M \rrbracket \circ \eta_x \circ \langle Id, \llbracket N \rrbracket \rangle =$ by Lemma 1 $= \llbracket M[N/x] \rrbracket$. □

We aim to prove that our model is *sensible* w.r.t. the operational semantics: a $\lambda_{+\|}$-term M has a non-empty interpretation if, and only if, M is solvable.

We start showing that the interpretation of every solvable term is non-empty (for the converse we will adapt Krivine's realizability method [13], see Section 5). This is an immediate corollary of the following propositions stating that the interpretation of a $\lambda_{+\|}$-term includes the union of the interpretations of its multiple hnf's and that the interpretation of any hnf is non-empty.

Proposition 7. *For all $M \in \Lambda_{+\|}$, we have $(\bigoplus_{m \in H(M)} (\bigodot_{N \in m} \llbracket N \rrbracket)) \subseteq \llbracket M \rrbracket$.*

Proof. It is enough to show that $(\bigoplus_{m \in H_n(M)} (\bigodot_{N \in m} \llbracket N \rrbracket)) \subseteq \llbracket M \rrbracket$ holds for all $n \in \mathbb{N}$; we prove it by induction on n. The case $n = 0$ is trivial. The proof of the inductive step goes by case analysis on the head subterm M' of $M \equiv \lambda \vec{z}.M'\vec{P}$.

- The case $M' \equiv x$ is trivial, and the case $M' \equiv \lambda y.Q$ is settled by Lemma 2.
- If $M' \equiv Q_1 \| Q_2$, we start by observing that $\llbracket M \rrbracket = \llbracket \lambda \vec{z}.Q_1\vec{P} \rrbracket \odot \llbracket \lambda \vec{z}.Q_2\vec{P} \rrbracket$. This is an easy consequence of the right distributivity of $\bullet$ over $\odot$ (Proposition 5) and of the fact that, by Proposition 6(*iii*), we have $\llbracket \lambda \vec{x}.(R_1 \| R_2) \rrbracket = \llbracket \lambda \vec{x}.R_1 \rrbracket \odot \llbracket \lambda \vec{x}.R_2 \rrbracket$, for all $\vec{x} \in \mathrm{Var}$ and $R_1, R_2 \in \Lambda_{+\|}$. Then, we can conclude by the inductive hypothesis.
- The case $M' \equiv Q_1 + Q_2$ is similar, and simpler, once noted that $\llbracket M \rrbracket = \llbracket \lambda \vec{z}.Q_1\vec{P} \rrbracket \oplus \llbracket \lambda \vec{z}.Q_2\vec{P} \rrbracket$ (again, by Proposition 5 and Proposition 6(*iii*)). □

We now show that every hnf has a non-empty interpretation.

Proposition 8. *For all $x, \vec{y} \in \mathrm{Var}$ and $\vec{Q} \in \Lambda_{+\|}$ we have $\llbracket \lambda \vec{y}.x\vec{Q} \rrbracket \neq \emptyset$.*

Proof. By Proposition 6(*iii*), it is sufficient to prove that, for all $x \in \mathrm{Var}$ and $\vec{Q} \in \Lambda_{+\|}$, we have $\llbracket x\vec{Q} \rrbracket \neq \emptyset$. To conclude, it is easy to show by induction on k that $([(x,\star)],\star) \in \llbracket xQ_1 \dots Q_k \rrbracket$. □

Theorem 2. *For all $M \in \Lambda_{+\|}$, if $H(M) \neq \emptyset$ then $\llbracket M \rrbracket \neq \emptyset$.*

Proof. Let $[N_1, \dots, N_k] \in H(M)$. By Proposition 7, $\bigodot_{1 \leq i \leq k} \llbracket N_i \rrbracket \subseteq \llbracket M \rrbracket$, and by Proposition 8 $\llbracket N_i \rrbracket \neq \emptyset$ for $1 \leq i \leq k$. We conclude that $\emptyset \neq \bigodot_{1 \leq i \leq k} \llbracket N_i \rrbracket \subseteq \llbracket M \rrbracket$. □

5 Saturated Sets and the Realizability Argument

In this section, we generalize Krivine's realizability technique [13] to $\lambda_{+\|}$-calculus and we use it for proving that $\lambda_{+\|}$-terms having a non-empty interpretation are all solvable. For notations and terminology, we mainly follow [3].

The saturation of a set S of terms expresses the fact that S is closed under weak head expansions. For the pure λ-calculus, this amounts to the well known condition of being closed under weak head β-expansion. For the extension of the λ-calculus we are dealing with, three cases of weak head expansions, corresponding to the possible shapes of the head term, must be considered.

Definition 8. *A set $S \subseteq \Lambda_{+\|}$ is* saturated *if the following conditions hold:*

- *if $M[N/x]\vec{P} \in S$ then $(\lambda x.M)N\vec{P} \in S$,*
- *if $(MQ\|NQ)\vec{P} \in S$ then $(M\|N)Q\vec{P} \in S$,*
- *if $M\vec{P} \in S$ and $N \in \Lambda_{+\|}$ then $(M+N)\vec{P} \in S$.*

We recall that the sets $\mathcal{N}_0, \mathcal{N}_1$ and $\mathcal{N}$ have been defined in Section 3.3. It is easy to check that $\mathcal{N}$ is saturated, whilst $\mathcal{N}_0$ is not. In the realizability argument, only saturated sets included within $\mathcal{N}_0$ and $\mathcal{N}$ will be considered.

Definition 9. *The set Sat_h of "small" saturated subsets of $\Lambda_{+\|}$ is defined by:*

$$Sat_h = \{S \subseteq \Lambda_{+\|} \mid \ S \text{ is saturated and } \mathcal{N}_0 \subseteq S \subseteq \mathcal{N}\}.$$

Given $A, B \subseteq \Lambda_{+\|}$, we define $A \to B = \{M \in \Lambda_{+\|} \mid (\forall N \in A)\ MN \in B\}$. The operator $\to$ is contravariant in its first argument and covariant in its second one, in other words, $A \to B \subseteq A' \to B'$ for all $A' \subseteq A$ and $B \subseteq B'$.

Lemma 3. $\mathcal{N}_0 \subseteq \Lambda_{+\|} \to \mathcal{N}_0 \subseteq \mathcal{N}_0 \to \mathcal{N} \subseteq \mathcal{N}$.

Proof. The first inclusion follows by definition, the second one is a consequence of the contravariance/covariance of the arrow. For the third one, it is enough to prove that, for all $M \in \Lambda_{+\|}$ and $x \in \mathrm{Var}$, $H(Mx) \neq \emptyset$ entails $H(M) \neq \emptyset$; this holds by Proposition 2(i). □

The set Sat_h enjoys the following closure properties.

Lemma 4. *The set Sat_h is closed under the arrow operator, finite unions, finite intersections, and under the map $\mathcal{F} : S \mapsto (\Lambda_{+\|} \to S)$.*

Proof. Given two sets $S_1, S_2 \in Sat_h$, it is straightforward to check that $S_1 \cap S_2$, $S_1 \cup S_2 \in Sat_h$ and that $S_1 \to S_2$ and $\Lambda_{+\|} \to S_2$ are saturated. The inclusions $\mathcal{N}_0 \subseteq S_1 \to S_2 \subseteq \mathcal{N}$ and $\mathcal{N}_0 \subseteq \Lambda_{+\|} \to S_2 \subseteq \mathcal{N}$ follow easily from Lemma 3 and contravariance/covariance of the arrow. □

We are going to define a function $(-)^\bullet : D \to Sat_h$, satisfying $(m :: \sigma)^\bullet = m^\bullet \to \sigma^\bullet$, where, for a multiset m of elements of D, $m^\bullet = \bigcap_{\alpha \in m} \alpha^\bullet$ and, in particular, $[]^\bullet = \Lambda_{+\|}$. Since $\star = [] :: \star$, the set $\star^\bullet$ must be a fixpoint of the function $\mathcal{F} : S \mapsto (\Lambda_{+\|} \to S)$. We now show that $\mathcal{N}_1$ is one of such fixpoints.

Proposition 9. *$\mathcal{N}_1 \in Sat_h$ and $\mathcal{N}_1 = \Lambda_{+\|} \to \mathcal{N}_1$.*

Proof. The saturation of $\mathcal{N}_1$ and the fact that $\mathcal{N}_0 \subseteq \mathcal{N}_1 \subseteq \mathcal{N}$ are both trivial. We now prove that $\mathcal{N}_1 = \Lambda_{+\|} \to \mathcal{N}_1$. Let $M \in \Lambda_{+\|} \to \mathcal{N}_1$. Since $M\Omega \in \mathcal{N}_1$, we get by Proposition 2(*ii*) that $M \in \mathcal{N}_1$. Conversely, let $M \in \mathcal{N}_1$ and $N \in \Lambda_{+\|}$. We conclude since, by Proposition 2(*iii*), we get $MN \in \mathcal{N}_1$. □

Observe that any element $\sigma \in D$ may be written in a unique way as $\sigma = \sigma_1 :: \cdots :: \sigma_n :: \star$, with $n \geq 0$ and $\sigma_n \neq []$ (and of course $\sigma_1, \ldots, \sigma_n$ have ranks strictly smaller than that of σ). This is called the *standard decomposition* of σ.

Definition 10. *Given $\sigma \in D$, we define $(\sigma)^\bullet \in Sat_h$ by induction on the rank k of σ. If $k = 0$, then $\sigma^\bullet = \star^\bullet = \mathcal{N}_1$. If $k > 0$ then $\sigma^\bullet = \sigma_1^\bullet \to \cdots \to \sigma_n^\bullet \to \mathcal{N}_1$, where $\sigma_1 :: \cdots :: \sigma_n :: \star$ is the standard decomposition of σ.*

Note that if $m \neq []$ or $\sigma \neq \star$, then the standard decomposition of $m :: \sigma$ is $m :: \sigma_1 :: \cdots :: \sigma_n :: \star$, where $\sigma_1 :: \cdots :: \sigma_n :: \star$ is the standard decomposition of σ. Hence, $(m :: \sigma)^\bullet = m^\bullet \to \sigma^\bullet$ holds in general, since $([] :: \star)^\bullet = \star^\bullet = \mathcal{N}_1 = \Lambda_{+\|} \to \mathcal{N}_1$.

We show now that the definition of $(-)^\bullet$ fits well with parallel composition.

Lemma 5. *Let $M, N \in \Lambda_{+\|}$, $\sigma = (\sigma_1, \sigma_2, \ldots), \tau = (\tau_1, \tau_2, \ldots) \in D$ and $\rho = \sigma \uplus \tau$. If $M \in \sigma^\bullet$ and $N \in \tau^\bullet$, then $M \| N \in \rho^\bullet$.*

Proof. Let $\rho_n :: \cdots :: \rho_1 :: \star$ be the standard decomposition of ρ. We have to show that $M \| N \in \rho_n^\bullet \to \cdots \to \rho_1^\bullet \to \mathcal{N}_1$. We prove it by induction on n.

If $n = 0$, then $\sigma = \tau = \rho = \star$. Hence, we conclude since $\star^\bullet = \mathcal{N}_1$ and $\mathcal{N}_1$ is closed under parallel composition.

If $n > 0$, then we have to show that, for all $Q \in \rho_n^\bullet$, $(M \| N)Q \in (\rho')^\bullet$ where $\rho' = \rho_{n-1} :: \cdots :: \rho_1 :: \star$. Since $M \in \sigma_1^\bullet$ and $N \in \tau_1^\bullet$, we have that $MQ \in (\sigma')^\bullet$ and $NQ \in (\tau')^\bullet$, where $\sigma' = (\sigma_2, \sigma_3, \ldots)$ and $\tau' = (\tau_2, \tau_3, \ldots)^\bullet$. Moreover, $\rho' = \sigma' \uplus \tau'$ and the standard decomposition of ρ' is strictly shorter than that of ρ. By the inductive hypothesis, we get $MQ \| NQ \in (\rho')^\bullet$. By saturation of $(\rho')^\bullet$, we conclude that $(M \| N)Q \in (\rho')^\bullet$, and hence $M \| N \in \rho^\bullet$. □

We are now able to prove the promised *adequacy lemma*, which constitutes the key tool in the realizability argument.

Definition 11. *A substitution $s = \{(x_1, N_1), \ldots, (x_k, N_k)\}$ is* adequate *for a multiset $m \in \mathcal{M}_f(D^{\mathrm{Var}})$ if:*

- *$m_x \neq []$ implies $x \in \{x_1, \ldots, x_k\}$, for all $x \in \mathrm{Var}$,*
- *$N_i \in m_{x_i}^\bullet$ for all $1 \leq i \leq k$.*

Observe that, if a substitution is adequate for some multiset $m \in \mathcal{M}_f(D^{\mathrm{Var}})$, then it is adequate for all submultisets of m.

Lemma 6. (Adequacy lemma) *Let $M \in \Lambda_{+\|}$, $(m, \sigma) \in [\![M]\!]$ and s be a substitution. If s is adequate for m, then $Ms \in \sigma^\bullet$.*

Proof. By structural induction on M.

- If $M \equiv x$, then $m = [(x, \sigma)]$ by Proposition 6(i). If s is adequate for m, then $(x, N) \in s$ for some $N \in [\sigma]^\bullet$. Hence, we have that $Ms = N \in [\sigma]^\bullet = \sigma^\bullet$.
- If $M \equiv PQ$, then by Proposition 6(ii), we have $m = m_0 \uplus m_1 \uplus \ldots \uplus m_k$ for some $k \geq 0$, and $\tau_1, \ldots, \tau_k \in D$ such that $(m_0, [\tau_1, \ldots, \tau_k] :: \sigma) \in [\![P]\!]$ and $(m_i, \tau_i) \in [\![Q]\!]$ *for* $1 \leq i \leq k$. Observe now that, if s is adequate for m then it is also adequate for $m_0, m_1, \ldots, m_k$, since they are all multisubsets of m. By the inductive hypothesis we have that:
  - $Ps \in ([\tau_1, \ldots, \tau_k] :: \sigma)^\bullet = [\tau_1, \ldots, \tau_k]^\bullet \to \sigma^\bullet$,
  - $Qs \in \tau_1^\bullet, \ldots, Qs \in \tau_k^\bullet$, which implies that $Qs \in [\tau_1, \ldots, \tau_k]^\bullet$.

  Hence, we can conclude that $(PQ)s \in \sigma^\bullet$.
- If $M \equiv \lambda x.P$, then by Proposition 6(iii), we have that $m = m'_{-x}$ and $\sigma = m'_x :: \sigma'$ for some $(m', \sigma') \in [\![P]\!]$. Let s be an adequate substitution for m'_{-x} and $Q \in (m'_x)^\bullet$. Since M is considered up to α-conversion, we can suppose without loss of generality that x does not occur in s. It is clear that $s' = s \cup \{(x, Q)\}$ is adequate for m' and hence, by the inductive hypothesis, we get $Ps' \in (\sigma')^\bullet$. Now we have that $Ps' = (Ps)[Q/x] \in (\sigma')^\bullet$ because x does not appear in s. Since $(\sigma')^\bullet$ is saturated and $(\lambda x.Ps) = (\lambda x.P)s$ we have that $(\lambda x.P)sQ \in (\sigma')^\bullet$. From the arbitrariness of $Q \in (m'_x)^\bullet$ we conclude that $(\lambda x.P)s \in (m'_x)^\bullet \to (\sigma')^\bullet = (m'_x :: \sigma')^\bullet$.
- If $M \equiv P + Q$, then (m, σ) belongs to, say, $[\![P]\!]$. Now, if s is adequate for m, then we get by the inductive hypothesis that $Ps \in \sigma^\bullet$ and we conclude, by saturation of $\sigma^\bullet$, that $(P + Q)s \in \sigma^\bullet$.
- If $M \equiv P \| Q$, then $m = m_1 \uplus m_2$ and $\sigma = \sigma_1 \bar{\uplus} \sigma_2$ with $(m_1, \sigma_1) \in [\![P]\!]$ and $(m_2, \sigma_2) \in [\![Q]\!]$. If s is adequate for m then it is also adequate for m_1, m_2 and, from the inductive hypothesis and Lemma 5, we conclude that $(P \| Q)s \in (\sigma_1 \bar{\uplus} \sigma_2)^\bullet$. □

Theorem 3. *For all* $M \in \Lambda_{+\|}$, *if* $[\![M]\!] \neq \emptyset$ *then* $M \in \mathcal{N}$.

Proof. Let $(m, \sigma) \in [\![M]\!]$. The substitution $s_{Id} = \{(x, x) \mid m_x \neq []\}$ is adequate for m (note that $\mathrm{Var} \subset \mathcal{N}_0$), and $Ms_{Id} = M$. Hence, by the adequacy lemma, we conclude that $M \in \sigma^\bullet \subseteq \mathcal{N}$. □

By Theorem 2 and Theorem 3 we finally get our main result.

Theorem 4. *For all* $M \in \Lambda_{+\|}$, $H(M) \neq \emptyset \Leftrightarrow [\![M]\!] \neq \emptyset$.

6 Conclusions and Further Work

We have defined a (relational) model $\mathcal{D}$ of a fairly standard parallel and non-deterministic extension of the pure λ-calculus, equipped with a notion of observation given by an operator H. In this framework, full abstraction spells out as follows: $(\forall M, N \in \Lambda_{+\|})[(\forall C[-])\ H(C[M]) \neq \emptyset \Rightarrow H(C[N]) \neq \emptyset]$ iff $[\![M]\!] \subseteq [\![N]\!]$. The "if" part of the previous statement (adequacy) is an easy consequence of Theorem 4. Nevertheless, the "only if" part fails. Indeed, given $\mathbf{I} \equiv \lambda x.x$, we have that $[\![\mathbf{I}]\!] = \{([], [\sigma] :: \sigma) \mid \sigma \in D)\}$ and $[\![\mathbf{I}\|\mathbf{I}]\!] = \{([], [\sigma, \sigma] :: (\sigma \bar{\uplus} \sigma)) \mid \sigma \in D)\}$. Hence, $[\![\mathbf{I}]\!] \not\subseteq [\![\mathbf{I}\|\mathbf{I}]\!]$ whilst it is not difficult to check that $\mathbf{I}$ and $\mathbf{I}\|\mathbf{I}$ are not separable using contexts. As suggested by the counterexample, the next step towards

full abstraction should be to enrich the syntax of the language by some "resource sensitive" operator, to increase the discriminating power of contexts.

Finally, we already know from [15, Sec. 3.3] that the theory induced on the pure untyped λ-calculus by our model $\mathcal{D}$ is $\mathcal{H}^*$ (just as the theory induced by Scott's $\mathcal{D}_\infty$); it would be interesting to generalize such a result to the extended setting, as a step in the study and classification of $\lambda_{+\|}$-theories, and models.

References

1. Asperti, A., Longo, G.: Categories, types and structures. Category theory for the working computer scientist. MIT Press, Cambridge (1991)
2. Barendregt, H.P.: The lambda calculus: Its syntax and semantics. North-Holland Publishing Co., Amsterdam (1984)
3. Berline, C.: From computation to foundations via functions and application: The λ-calculus and its webbed models. Theor. Comp. Sci. 249, 81–161 (2000)
4. Boudol, G.: Lambda-calculi for (strict) parallel functions. Inf. Comput. 108(1), 51–127 (1994)
5. Boudol, G., Lavatelli, C.: Full abstraction for lambda calculus with resources and convergence testing. In: Kirchner, H. (ed.) CAAP 1996. LNCS, vol. 1059, pp. 302–316. Springer, Heidelberg (1996)
6. Bucciarelli, A., Ehrhard, T., Manzonetto, G.: Not enough points is enough. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 268–282. Springer, Heidelberg (2007)
7. Danos, V., Krivine, J.-L.: Disjunctive tautologies as synchronisation schemes. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 292–301. Springer, Heidelberg (2000)
8. Dezani Ciancaglini, M., de Liguoro, U., Piperno, A.: Filter models for conjunctive-disjunctive lambda-calculi. Theor. Comput. Sci. 170(1-2), 83–128 (1996)
9. Dezani Ciancaglini, M., de Liguoro, U., Piperno, A.: A filter model for concurrent λ-calculus. SIAM J. Comput. 27(5), 1376–1419 (1998)
10. Ehrhard, T.: Hypercoherences: a strongly stable model of linear logic. Math. Struct. Comp. Sci. 3(4), 365–385 (1993)
11. Faure, G., Miquel, A.: A categorical semantics for the parallel lambda-calculus (submitted), `http://rho.loria.fr/data/lics07.pdf`
12. Girard, J.-Y.: Linear Logic. Theor. Comp. Sci. 50 (1988)
13. Krivine, J.-L.: Lambda-calculus. Types and models. Ellis Horwood, Hemel Hempstead (1993)
14. Laird, J.: Bidomains and full abstraction for countable nondeterminism. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 352–366. Springer, Heidelberg (2006)
15. Manzonetto, G.: Models and theories of lambda calculus. Ph.D. Thesis, Univ. Ca'Foscari (Venezia), Univ. Paris 7, Paris (2008)
16. Ong, C.-H.L.: Non-determinism in a functional setting. In: Proc. of LICS 1993, pp. 275–286 (1993)
17. Paolini, L.: A stable programming language. Inf. Comput. 204(3), 339–375 (2006)
18. Plotkin, G.D.: A powerdomain construction. SIAM J. Comput. 5(3), 452–487 (1976)
19. Plotkin, G.D.: LCF considered as a programming language. Theor. Comput. Sci. 5(3), 225–255 (1977)
20. Selinger, P.: The λ-calculus is algebraic. J. Funct. Program. 12(6), 549–566 (2002)

The NP-Completeness of Reflected Fragments of Justification Logics

Samuel R. Buss[1,⋆] and Roman Kuznets[2,⋆⋆]

[1] Department of Mathematics, University of California, San Diego
La Jolla, CA 92093-0112, USA
sbuss@ucsd.edu
[2] Institut für Informatik und angewandte Mathematik, Universität Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
kuznets@iam.unibe.ch

Abstract. Justification Logic studies epistemic and provability phenomena by introducing justifications/proofs into the language in the form of justification terms. Pure justification logics serve as counterparts of traditional modal epistemic logics, and hybrid logics combine epistemic modalities with justification terms. The computational complexity of pure justification logics is typically lower than that of the corresponding modal logics. Moreover, the so-called reflected fragments, which still contain complete information about the respective justification logics, are known to be in NP for a wide range of justification logics, pure and hybrid alike. This paper shows that, under reasonable additional restrictions, these reflected fragments are NP-complete, thereby proving a matching lower bound.

1 Introduction and Main Definitions

Justification Logic is an emerging field that studies provability, knowledge, and belief via explicit proofs or justifications that are part of the language. A justification logic is essentially a refined analogue of a modal epistemic logic. Whereas a modal epistemic logic uses the formula $\Box F$ to indicate that F is known to be true, a justification logic uses $t : F$ instead, where t is a term that describes a 'justification' or proof of F. This construction allows justification logics to reason about both formulas and proofs at the same time, avoiding the need to treat provability at the metalevel.

Because Justification Logic can reason directly about explicit proofs, it provides more concrete and constructive analogues of modal epistemic logics. For example, the modal distribution axiom $\Box(F \to G) \to (\Box F \to \Box G)$ is replaced in Justification Logic by the axiom $s : (F \to G) \to (t : F \to (s \cdot t) : G)$. The latter replaces the distribution axiom with a computationally explicit construction. Justification logics are very promising for structural proof theory and have already

⋆ Partially supported by NSF grant DMS–0700533.
⋆⋆ Supported by Swiss National Science Foundation grant 200021–117699. The initial stages of the research were partially supported by a CUNY Graduate Center Research Grant for Doctoral Students.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 122–136, 2009.

proven to be fruitful in finding new approaches to common knowledge ([Art06]) and Logical Omniscience Problem ([AK06]). For further discussion on the various applications of Justification Logic, see [Art08b].

The goal of the present paper is to prove the NP-hardness of the Derivability Problem for the reflected fragments of justification logics, matching the already known upper bound. We begin by reviewing some definitions of justification logics.

The first justification logic, the Logic of Proofs LP, was introduced by Artemov [Art95] to provide a provability semantics for the modal logic S4 (see also [Art01]). The language of LP

$$F ::= p \mid \bot \mid (F \to F) \mid t{:}F \ ,$$
$$t ::= x \mid c \mid (t \cdot t) \mid (t + t) \mid !\, t$$

contains an additional operator $t{:}F$, read 'term t serves as a justification/proof of formula F.' Here p stands for a sentence letter, x for a justification variable, and c for a justification constant.

Statements $t{:}F$ can be seen as refinements of modal statements $\Box F$ because the latter say that F is known whereas the former additionally provide a rationale for such knowledge. This relationship is demonstrated through the recursively defined operation of *forgetful projection* that maps justification formulas to modal formulas: $(t{:}F)^\circ = \Box(F^\circ)$, and commutes with Boolean connectives: $(F \to G)^\circ = F^\circ \to G^\circ$, where $p^\circ = p$ and $\bot^\circ = \bot$.

Axioms and rules of LP:

A1. A complete axiomatization of classical propositional logic by finitely many axiom schemes; rule *modus ponens*

A2. *Application Axiom* $s{:}(F \to G) \to (t{:}F \to (s \cdot t){:}G)$

A3. *Monotonicity Axiom* $s{:}F \to (s + t){:}F, \quad t{:}F \to (s + t){:}F$

A4. *Factivity Axiom* $t{:}F \to F$

A5. *Positive Introspection Axiom* $t{:}F \to !\, t{:}t{:}F$

R4. *Axiom Internalization Rule*: $\dfrac{}{c{:}A}$

where A is an axiom and c is a justification constant

LP is the exact counterpart of S4 (note the similarity of their axioms): namely, let $X^\circ = \{F^\circ \mid F \in X\}$ for a set X of justification formulas and let LP be identified with the set of its theorems, then

Theorem 1 (Realization Theorem, [Art95, Art01]). $\mathsf{LP}^\circ = \mathsf{S4}$.

For some applications (e.g., to avoid Logical Omniscience [AK06] or to study self-referentiality [Kuz08c]) the use of constants needs to be restricted; this is achieved using *constant specifications*. A *constant specification* $\mathcal{CS}$ is a set of instances of rule R4:

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

Given a constant specification $\mathcal{CS}$, the logic $\mathsf{LP}_{\mathcal{CS}}$ is the result of replacing R4 in LP by its relativized version:

R4$_{\mathcal{CS}}$. *Relativized Axiom Internalization Rule*: $\dfrac{c{:}A \in \mathcal{CS}}{c{:}A}$

For the Realization Theorem to hold, i.e., for $(\mathsf{LP}_{\mathcal{CS}})^\circ = \mathsf{S4}$, it is necessary and sufficient that $\mathcal{CS}$ be *axiomatically appropriate*:

Definition. A constant specification $\mathcal{CS}$ is called:

- *axiomatically appropriate*[1] if every axiom is justified by at least one constant;
- *schematic*[2] if each constant justifies several (maybe 0) axiom schemes and only them;
- *schematically injective*[3] if it is schematic and each constant justifies no more than one axiom scheme.

Whereas it is well known that the Derivability Problem for $\mathsf{S4}$ is PSPACE-complete ([Lad77]), it was shown in [Kuz00] that the same problem for $\mathsf{LP}_{\mathcal{CS}}$ is in Π_2^p for any schematic $\mathcal{CS}$ (we always assume $\mathcal{CS}$ to be polynomial time decidable); in particular, LP itself is in Π_2^p. Milnikel in [Mil07] proved a matching lower bound, the Π_2^p-hardness of $\mathsf{LP}_{\mathcal{CS}}$ under the assumption that $\mathcal{CS}$ is axiomatically appropriate and schematically injective.

The so-called *reflected fragment* rLP of the Logic of Proofs was first studied by N. Krupski in [Kru03] (see also [Kru06]):

Definition. For any justification logic $\mathsf{JL}_{\mathcal{CS}}$ with a constant specification $\mathcal{CS}$, its reflected fragment is

$$\mathsf{rJL}_{\mathcal{CS}} = \{t{:}F \mid \mathsf{JL}_{\mathcal{CS}} \vdash t{:}F\} \ .$$

We will write $\mathsf{rJL}_{\mathcal{CS}} \vdash t{:}F$ to mean $t{:}F \in \mathsf{rJL}_{\mathcal{CS}}$.

The reflected fragment bears complete information about the logic as the following theorem shows:

Theorem 2 ([Kru03, Kru06]). *For any axiomatically appropriate $\mathcal{CS}$,*

$$\mathsf{LP}_{\mathcal{CS}} \vdash F \qquad \Longleftrightarrow \qquad (\exists t)\mathsf{rLP}_{\mathcal{CS}} \vdash t{:}F \ .$$

The $\Longrightarrow$-direction constitutes the Constructive Necessitation Property (for details, see [Art01]); the $\Longleftarrow$-direction easily follows from Factivity Axiom A4.

Theorem 3 ([Kru03, Kru06]). *For any schematic $\mathcal{CS}$, the Derivability Problem for $\mathsf{rLP}_{\mathcal{CS}}$ is in NP.*

To prove this theorem, N. Krupski developed an independent axiomatization for $\mathsf{rLP}_{\mathcal{CS}}$ that we will call the $*$-calculus.

[1] The term is due to Fitting.
[2] The term is due to Milnikel although the idea goes back to Mkrtychev.
[3] The term is due to Milnikel.

Axioms and rules of the $*$-calculus:

$*\mathcal{CS}$. For any $c{:}A \in \mathcal{CS}$ $\qquad$ axiom $c{:}A$

$*$A2. *Application Rule* $\qquad \dfrac{s{:}(F \to G) \qquad t{:}F}{s \cdot t{:}G}$

$*$A3. *Sum Rule* $\qquad \dfrac{s{:}F}{s+t{:}F} \qquad \dfrac{t{:}F}{s+t{:}F}$

$*$A5. *Positive Introspection Rule* $\qquad \dfrac{t{:}F}{!t{:}t{:}F}$

In this paper, we prove the matching lower bound for $\mathsf{rLP}_{\mathcal{CS}}$, namely that the Derivability Problem for $\mathsf{rLP}_{\mathcal{CS}}$ is NP-complete. The proof is by a many-one polynomial-time reduction from a known NP-complete problem, the Vertex Cover problem. As in Milnikel's lower bound for $\mathsf{LP}_{\mathcal{CS}}$, we have to impose the additional restriction that $\mathcal{CS}$ is axiomatically appropriate and schematically injective.

The paper is structured as follows. Section 2 defines a coding of a graph by propositional formulas and shows how the existence of a vertex cover can be described in terms of these formulas. Section 3 develops justification terms that encode several standard methods of propositional reasoning. Although the formulas that describe the existence of a vertex cover depend on the cover itself rather than only on its size, Sect. 4 shows how to eliminate this dependency by using the terms from Sect. 3 to encode particular derivations of the formulas from Sect. 2. Section 5 finishes the proof of the polynomial-time reduction. Section 6 discusses extending this result to other justification logics.

2 Graph Coding and Preliminaries

A graph $G = \langle V, E\rangle$ has a finite set V of vertices and a finite set E of undirected edges. We assume w.l.o.g. that $V = \{1, \ldots, N\}$ for some N, and we represent an edge e between vertices k and l as the set $e = \{k, l\}$ with the endpoints denoted by $v_1(e) < v_2(e)$. A vertex cover for G is a set C of vertices such that each edge $e \in E$ has at least one endpoint in C. The Vertex Cover problem is the problem of, given a graph G and an $L \geq 0$, determining if G has a vertex cover of size $\leq L$. The Vertex Cover problem is one of the classic NP-complete problems.

We define below formulas F_V, F_C, and F_G that will help build a many-one reduction from Vertex Cover to $\mathsf{rLP}_{\mathcal{CS}}$. These formulas will include large conjunctions. To avoid the dependence of the $\mathsf{LP}_{\mathcal{CS}}$-derivations on the vertex cover, we will use balanced conjunctions (see [BB93]):

Definition. Each formula is a *balanced conjunction of depth* 0. If A and B are both balanced conjunctions of depth k, then $A \wedge B$ is a *balanced conjunction of depth* $k+1$.

Clearly, a balanced conjunction of depth k is also a balanced conjunction of depth l for any $0 \leq l \leq k$. Thus, we are mainly interested in how deeply a

given formula is conjunctively balanced. For any conjunction $C_1 \wedge \cdots \wedge C_{2^k}$ of 2^k formulas, we assume that the omitted parentheses are such that the resulting balanced conjunction has the maximal possible depth, i.e., depth $\geq k$.

We also need to refer to C_i's that form $F = C_1 \wedge \cdots \wedge C_{2^k}$. The following inductive definition of *depth k conjuncts*, or simply *k-conjuncts*, generalizes the definition of *conjuncts* in an ordinary conjunction:

Definition. Each formula is a 0-*conjunct* of itself. If $C \wedge D$ is a k-conjunct of formula F, then C and D are both $(k+1)$-*conjuncts* of F.

For instance, the conjuncts of an ordinary conjunction are its 1-conjuncts; all C_i's in $C_1 \wedge \cdots \wedge C_{2^k}$ are its k-conjuncts. More generally, any balanced conjunction of depth k must have exactly 2^k occurrences of k-conjuncts (with possibly several occurrences of the same formula).

To make full use of balanced conjunctions, it is convenient to restrict attention to instances of the Vertex Cover problem for graphs in which both the number of vertices and the number of edges are powers of 2. These are called *binary exponential graphs*. It is also helpful to only consider vertex covers whose size is a power of 2; these we call *binary exponential* vertex covers. Fortunately, the version of the Vertex Cover (VC) problem restricted to binary exponential graphs and their binary exponential vertex covers is also NP-complete:

Theorem 4. *The Binary Vertex Cover (BVC) problem of determining for a given binary exponential graph G and a given $l \geq 0$ whether G has a vertex cover of size $\leq 2^l$ is NP-complete.*

Proof. Since BVC is an instance of the standard VC problem, and since VC is NP-complete, it suffices to construct a polynomial-time many-one reduction from VC to BVC. Suppose we are given an instance of VC; namely, we are given a graph G_0 and an integer L and wish to determine if G_0 has a vertex cover of size $\leq L$. We give a polynomial time procedure that constructs a binary exponential graph G and a value l so that G_0 has a vertex cover of size $\leq L$ iff G has a vertex cover of size $\leq 2^l$. The graph G is constructed in three stages; each stage causes only a constant factor increase in the size of the graph.

Stage 1. Increasing the size of the vertex cover. Let $0 \leq L' < L$ such that $L + L' = 2^l - 1$ for some integer $l \geq 0$. The graph $G' = \langle V', E' \rangle$ is obtained from G_0 by adding $2L'$ new vertices broken into L' disjoint pairs with the vertices in each pair joined by a new edge (L' new edges overall). G_0 has a vertex cover of size $\leq L$ iff G' has a vertex cover of size $\leq 2^l - 1$.

Stage 2. Increasing the number of edges. Choose integer $0 < M'' \leq |E'|$ such that $|E'| + M'' = 2^m$ for some integer $m \geq 0$. The graph $G'' = \langle V'', E'' \rangle$ is obtained by adding $M'' + 1$ new vertices to G' with one of these vertices joined to all M'' others (M'' new edges overall). G' has a vertex cover of size $\leq 2^l - 1$ iff G'' has a vertex cover of size $\leq 2^l$.

Stage 3. Increasing the number of vertices. Choose integer $0 \leq N''' < |V''|$ such that $|V''| + N''' = 2^n$ for some integer $n \geq 0$. The graph $G = G'''$ is obtained by

adding N''' isolated vertices to G''. G'' has a vertex cover of size $\leq 2^l$ iff G''' has a vertex cover of size $\leq 2^l$.

It is clear from the construction that G is a binary exponential graph such that G_0 has a vertex cover of size $\leq L$ iff G has a vertex cover of size $\leq 2^l$. □

Definition. Let $G = \langle V, E\rangle$ be a binary exponential graph with edge set $E = \{e_1, \ldots, e_{2^m}\}$. Let $C = \{i_1, i_2, \ldots, i_{2^l}\} \subseteq V$ be a possible binary exponential vertex cover for G, where $i_1 < i_2 < \cdots < i_{2^l}$. We define the following formulas:

a. $F_C = p_{i_1} \wedge \cdots \wedge p_{i_{2^l}}$.
b. For each edge $e = \{k, l\}$, where $k < l$, $F_e = p_k \vee p_l = p_{v_1(e)} \vee p_{v_2(e)}$.
c. $F_G = F_{e_1} \wedge \cdots \wedge F_{e_{2^m}}$.

The proof of the following properties of the translation is an easy exercise ($\vdash$ denotes derivability in classical propositional logic):

Lemma 5. *For any binary exponential graph $G = \langle V, E\rangle$ and any binary exponential set $C \subseteq V$,*
1. $\vdash F_V \to F_G$;
2. $\vdash F_V \to F_C$;
3. $\vdash F_C \to F_G$ *iff* *C is a vertex cover for G.*

Our goal is to reduce BVC to derivability in $\mathsf{rLP}_{\mathcal{CS}}$ for a certain class of $\mathcal{CS}$. To this end, we take a particular derivation of $F_V \to F_G$ that proceeds by first proving $F_V \to F_C$, followed by an attempt at a proof of $F_C \to F_G$ that succeeds iff C is a vertex cover. Finally, hypothetical syllogism (HS) is applied to infer $F_V \to F_G$. We further encode this derivation as a justification term t so that $\mathsf{rLP}_{\mathcal{CS}} \vdash t : (F_V \to F_G)$ iff C is a vertex cover. In BVC we need to determine whether there exists a vertex cover of (at most) a given size rather than whether a given set of vertices is a vertex cover. Thus, $t : (F_V \to F_G)$ should not depend on C but may (and should) depend on the size of C. Since C has already been "syllogized away" from formula $F_V \to F_G$, it remains to make sure that term t only depends on the size of C. Although the derivations of $F_V \to F_C$ and $F_C \to F_G$ have C explicitly present in them, the terms encoding them, and therefore t, can be made independent of C. This is the main reason why we use balanced conjunctions: this way all k-conjuncts are interchangeable.

Note about the use of constants. Throughout the paper, the minimum requirement on $\mathcal{CS}$ would be axiomatic appropriateness and schematicness. As a consequence, we can always assume that for any axiom scheme there exists a constant justifying it. So it makes sense to choose one such constant for each axiom scheme. The list of names for these fixed constants along with the corresponding axiom schemes consistently used in the paper can be found below:

$$
\begin{array}{lll}
\mathsf{rLP}_{\mathcal{CS}} \vdash c_1 & : & (X \to (Y \to X)) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_2 & : & ((X \to (Y \to Z)) \to ((X \to Y) \to (X \to Z))) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\wedge 1} & : & (X \wedge Y \to X) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\wedge 2} & : & (X \wedge Y \to Y) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\wedge} & : & (X \to (Y \to X \wedge Y)) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\vee 1} & : & (X \to X \vee Y) \\
\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\vee 2} & : & (Y \to X \vee Y)
\end{array}
$$

Note that we have assumed that certain axiom schemes are present among the propositional axioms chosen for A1. The beginning of Sect. 5 discusses why this assumption is not essential.

3 Justification Terms Encoding Propositional Reasoning

For all lemmas in the section, schematicness and axiomatic appropriateness are sufficient for the $\Longleftarrow$-direction; schematic injectivity is required for the $\Longrightarrow$-direction only.

The size of terms is defined in a standard way: $|c| = |x| = 1$ for any constant and any variable, $|(t \cdot s)| = |(t + s)| = |t| + |s| + 1$, $|!\,t| = |t| + 1$.

Lemma 6 (Encoding the Hypothetical Syllogism Rule). *The operation*

$$\mathrm{syl}(t, s) = \big(c_2 \cdot (c_1 \cdot s)\big) \cdot t$$

with $|\mathrm{syl}(t, s)| = |t| + |s| + 5$ *encodes the Hypothetical Syllogism Rule, i.e.,*

$$\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{syl}(t, s) : H \iff \begin{array}{l} H = A \to C \text{ such that for some } B \\ \mathsf{rLP}_{\mathcal{CS}} \vdash t : (A \to B) \quad \text{and} \quad \mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C). \end{array}$$

Proof. ($\Longleftarrow$). Here is a derivation of $t : (A \to B), s : (B \to C) \vdash \mathrm{syl}(t, s) : (A \to C)$:

$$\begin{array}{rcll} c_1 & : & ((B \to C) \to (A \to (B \to C))) & (*\mathcal{CS}) \\ s & : & (B \to C) & (\text{Hyp}) \\ c_1 \cdot s & : & (A \to (B \to C)) & (*\text{A2}) \\ c_2 & : & ((A \to (B \to C)) \to ((A \to B) \to (A \to C))) & (*\mathcal{CS}) \\ c_2 \cdot (c_1 \cdot s) & : & ((A \to B) \to (A \to C)) & (*\text{A2}) \\ t & : & (A \to B) & (\text{Hyp}) \\ (c_2 \cdot (c_1 \cdot s)) \cdot t & : & (A \to C) & (*\text{A2}) \end{array}$$

($\Longrightarrow$). Consider an arbitrary derivation of $\mathrm{syl}(t, s) : H$ in the $*$-calculus. It can easily be seen that any such derivation must have the same structure as the one used for the $\Longleftarrow$-direction above: the only difference can be in the choice of axioms for constants c_1 and c_2 and of formulas for terms s and t. Since $\mathcal{CS}$ is schematically injective, we know the form of axioms proven by c_1 and c_2. Thus, we can shape this as a unification problem: find X_1, Y_1, X_2, Y_2, Z_2, X_s, and X_t such that $\mathsf{rLP}_{\mathcal{CS}} \vdash s : X_s$, $\mathsf{rLP}_{\mathcal{CS}} \vdash t : X_t$, and the following is a $*$-calculus derivation of $s : X_s, t : X_t \vdash \mathrm{syl}(t, s) : H$:

$$\begin{array}{rcll} 1. & c_1 : (X_1 \to (Y_1 \to X_1)) & (*\mathcal{CS}) \\ 2. & s : X_s & (\text{Hyp}) \\ 3. & c_1 \cdot s : (Y_1 \to X_1) & (*\text{A2}) \\ 4. & c_2 : ((X_2 \to (Y_2 \to Z_2)) \to ((X_2 \to Y_2) \to (X_2 \to Z_2))) & (*\mathcal{CS}) \\ 5. & c_2 \cdot (c_1 \cdot s) : ((X_2 \to Y_2) \to (X_2 \to Z_2)) & (*\text{A2}) \\ 6. & t : X_t & (\text{Hyp}) \\ 7. & (c_2 \cdot (c_1 \cdot s)) \cdot t : H & (*\text{A2}) \end{array}$$

To make the applications of rule $*$A2 work in lines 3, 5, and 7, the unification variables have to satisfy the following equations:

$$X_1 = X_s \qquad \text{from 3.} \tag{1}$$
$$X_2 \to (Y_2 \to Z_2) = Y_1 \to X_1 \qquad \text{from 5.} \tag{2}$$
$$X_2 \to Y_2 = X_t \qquad \text{from 7.} \tag{3}$$
$$X_2 \to Z_2 = H \qquad \text{from 7.} \tag{4}$$

By (1) and (2), $X_s = X_1 = Y_2 \to Z_2$. This equation combined with (3) and (4) shows that H is indeed an implication that follows by HS from X_t and X_s justified by t and s respectively. □

Lemma 7 (Stripping k conjunctions). *For any integer $k \geq 0$ there exists a term t_k of size $O(k)$ that encodes the operation of stripping k conjunctions, i.e.,*

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_k : D \qquad \Longleftrightarrow \qquad D = H \to C, \text{ where } C \text{ is a } k\text{-conjunct of } H.$$

Proof. We prove by induction on k that the conditions are satisfied for

$$t_0 = (c_2 \cdot c_1) \cdot c_1 \ ,$$
$$t_{k+1} = \mathrm{syl}(c_{\wedge 1} + c_{\wedge 2}, t_k) \ .$$

It is clear that $|t_k| = 8k + 5$ because $|t_0| = 5$ and $|t_{k+1}| = |t_k| + 8$.

Base case, $k = 0$. ($\Longleftarrow$). If C is a 0-conjunct of H, then $H = C$, and it is easy to see that t_0 corresponds to the standard derivation of the tautology $C \to C$.
($\Longrightarrow$). Any $*$-derivation of $t_0 : D$ must have the form:

1.	$c_2 : ((X_2 \to (Y_2 \to Z_2)) \to ((X_2 \to Y_2) \to (X_2 \to Z_2)))$	($*\mathcal{CS}$)
2.	$c_1 : (X_1 \to (Y_1 \to X_1))$	($*\mathcal{CS}$)
3.	$c_2 \cdot c_1 : ((X_2 \to Y_2) \to (X_2 \to Z_2))$	($*$A2)
4.	$c_1 : (X_3 \to (Y_3 \to X_3))$	($*\mathcal{CS}$)
5.	$(c_2 \cdot c_1) \cdot c_1 : D$	($*$A2)

For $*$A2 from line 5 to be valid, it is necessary that $D = X_2 \to Z_2$. It follows from $*$A2 in line 3 that $X_2 \to (Y_2 \to Z_2) = X_1 \to (Y_1 \to X_1)$, in which case $X_2 = X_1 = Z_2$. Therefore, $D = X_2 \to X_2$, which is an implication from a formula to its 0-conjunct.

Induction step. ($\Longleftarrow$). Let H be a formula with a $(k+1)$-conjunct C. Then H must be of the form $H_1 \wedge H_2$ with C being a k-conjunct of H_i for some $i = 1, 2$. By the induction hypothesis, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_k : (H_i \to C)$ for this i. Since, in addition, $\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\wedge 1} : (H \to H_1)$ and $\mathsf{rLP}_{\mathcal{CS}} \vdash c_{\wedge 2} : (H \to H_2)$, by rule $*$A3, $\mathsf{rLP}_{\mathcal{CS}} \vdash (c_{\wedge 1} + c_{\wedge 2}) : (H \to H_i)$ for both $i = 1$ and $i = 2$. Then, by Lemma 6, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k+1} : (H \to C)$.
($\Longrightarrow$). By the induction hypothesis, t_k justifies only implications from a formula to one of its k-conjuncts. It is clear from rule $*$A3 that $c_{\wedge 1} + c_{\wedge 2}$ justifies only implications from a formula to one of its 1-conjuncts. By Lemma 6, t_{k+1} justifies only hypothetical syllogisms obtained from the latter and the former, but a k-conjunct of a 1-conjunct of a formula is its $(k+1)$-conjunct. □

Lemma 8. *For any term s and any integer $l \geq 0$ there exists a term* $\mathrm{conj}(s,l)$ *of size* $O(|s|2^l)$ *with the following property:*

$$\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{conj}(s,l) : D \qquad \Longleftrightarrow \qquad \begin{array}{l} D = B \to C_1 \wedge \cdots \wedge C_{2^l} \text{ such that} \\ \mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C_i) \text{ for all } i = 1, \ldots, 2^l. \end{array}$$

Proof. We prove by induction on l that the conditions are satisfied for

$$\begin{aligned} \mathrm{conj}(s,0) &= \mathrm{syl}(s, t_0) \ , \\ \mathrm{conj}(s, l+1) &= \Big(c_2 \cdot \mathrm{syl}\big(\mathrm{conj}(s,l), c_\wedge\big) \Big) \cdot \mathrm{conj}(s,l) \ . \end{aligned}$$

It is not hard to see that $|\mathrm{conj}(s,l)| = 2^l(|s|+19)-9$ because $|\mathrm{conj}(s,0)| = |s|+10$ and $|\mathrm{conj}(s,l+1)| = 2|\mathrm{conj}(s,l)| + 9$.

Base case, $l = 0$. ($\Longleftarrow$). For any C, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_0 : (C \to C)$ by Lemma 7. Then, by Lemma 6, $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C)$ implies $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{syl}(s, t_0) : (B \to C)$.
($\Longrightarrow$). By Lemma 6, $\mathrm{syl}(s, t_0)$ justifies only implications $B \to C$ for which there exists an A such that $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to A)$ and $\mathsf{rLP}_{\mathcal{CS}} \vdash t_0 : (A \to C)$. By Lemma 7, the latter implies $A = C$. Therefore, $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C)$.[4]

Induction step. ($\Longleftarrow$). Let $H = C_1 \wedge \cdots \wedge C_{2^{l+1}}$ with $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C_i)$ for all its $(l+1)$-conjuncts. Then $H = H_1 \wedge H_2$ where $C_1, C_2, \ldots, C_{2^l}$ are l-conjuncts of H_1 and $C_{2^l+1}, C_{2^l+2}, \ldots, C_{2^{l+1}}$ are l-conjuncts of H_2. By the induction hypothesis,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{conj}(s,l) : (B \to H_1) \ , \tag{5}$$
$$\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{conj}(s,l) : (B \to H_2) \ . \tag{6}$$

In addition, $\mathsf{rLP}_{\mathcal{CS}} \vdash c_\wedge : (H_1 \to (H_2 \to H_1 \wedge H_2))$; in other words,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash c_\wedge : (H_1 \to (H_2 \to H)) \ . \tag{7}$$

From (7) and (5) by Lemma 6, for $s' = \mathrm{syl}(\mathrm{conj}(s,l), c_\wedge)$ we have

$$\mathsf{rLP}_{\mathcal{CS}} \vdash s' : (B \to (H_2 \to H)) \ .$$

Then, from (6) and $\mathsf{rLP}_{\mathcal{CS}} \vdash c_2 : ((B \to (H_2 \to H)) \to ((B \to H_2) \to (B \to H)))$:

$$\begin{array}{ll} \mathsf{rLP}_{\mathcal{CS}} \vdash c_2 \cdot s' : ((B \to H_2) \to (B \to H)) & \text{and, finally,} \\ \mathsf{rLP}_{\mathcal{CS}} \vdash (c_2 \cdot s') \cdot \mathrm{conj}(s,l) : (B \to H) \ . & \end{array}$$

It remains to note that $\mathrm{conj}(s, l+1) = (c_2 \cdot s') \cdot \mathrm{conj}(s,l)$.
($\Longrightarrow$). By Lemma 6, the rule

$$\frac{t : (A \to B) \quad s : (B \to C)}{\mathrm{syl}(t,s) : (A \to C)} (\mathrm{Syl})$$

[4] Note that, in general, $\mathrm{conj}(s,0) = s$ does not satisfy the $\Longrightarrow$-direction.

is admissible in the $*$-calculus. So any $*$-derivation of $\mathrm{conj}(s, l+1) : D$ must contain the following key elements (we have already incorporated the induction hypothesis about $\mathrm{conj}(s, l)$ as well as Lemma 6):

$$\begin{array}{lrl}
1. & \mathrm{conj}(s,l) : (B \to C_1 \wedge C_2 \wedge \cdots \wedge C_{2^l}) & \text{(IH)} \\
2. & c_\wedge : (X_\wedge \to (Y_\wedge \to X_\wedge \wedge Y_\wedge)) & (*\mathcal{CS}) \\
3. & s' : (B \to (Y_\wedge \to X_\wedge \wedge Y_\wedge)) & \text{(Syl)} \\
4. & c_2 : ((X_2 \to (Y_2 \to Z_2)) \to ((X_2 \to Y_2) \to (X_2 \to Z_2))) & (*\mathcal{CS}) \\
5. & c_2 \cdot s' : ((X_2 \to Y_2) \to (X_2 \to Z_2)) & (*\text{A2}) \\
6. & \mathrm{conj}(s,l) : (B' \to C_{2^l+1} \wedge C_{2^l+2} \wedge \cdots \wedge C_{2^{l+1}}) & \text{(IH)} \\
7. & (c_2 \cdot s') \cdot \mathrm{conj}(s,l) : D & (*\text{A2})
\end{array}$$

where $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C_i)$ and $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B' \to C_{2^l+i})$ for $i = 1, \ldots, 2^l$. Let us collect all unification equations necessary for this to be a valid fragment of a $*$-derivation:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_{2^l} = X_\wedge \qquad \text{from 3.} \qquad (8)$$

$$B \to (Y_\wedge \to X_\wedge \wedge Y_\wedge) = X_2 \to (Y_2 \to Z_2) \qquad \text{from 5.} \qquad (9)$$

$$B' \to C_{2^l+1} \wedge C_{2^l+2} \wedge \cdots \wedge C_{2^{l+1}} = X_2 \to Y_2 \qquad \text{from 7.} \qquad (10)$$

$$X_2 \to Z_2 = D \qquad \text{from 7.} \qquad (11)$$

By (9) and (10), $B = X_2 = B'$. Thus, $\mathsf{rLP}_{\mathcal{CS}} \vdash s : (B \to C_i)$ for $i = 1, \ldots, 2^{l+1}$. Also

$$Y_\wedge = Y_2 = C_{2^l+1} \wedge C_{2^l+2} \wedge \cdots \wedge C_{2^{l+1}} \ ,$$

again by (9) and (10). So, by (8) and (9),

$$Z_2 = X_\wedge \wedge Y_\wedge = (C_1 \wedge C_2 \wedge \cdots \wedge C_{2^l}) \wedge (C_{2^l+1} \wedge C_{2^l+2} \wedge \cdots \wedge C_{2^{l+1}}) \ .$$

By (11), D is indeed an implication from B to this balanced conjunction for all of whose $(l+1)$-conjuncts term s justifies their entailment from B. □

Lemma 9. *For the term* $\mathrm{disj} = c_{\vee 1} + c_{\vee 2}$ *of size* $O(1)$,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{disj} : D \qquad \Longleftrightarrow \qquad D = B \to H, \text{ where } B \text{ is a disjunct of } H.$$

Proof. Easily follows from $*$A3 and $*\mathcal{CS}$. □

4 Reduction from Vertex Cover, Part I

We now use the justification terms from the previous section to build a polynomial-time many-one reduction from BVC to $\mathsf{rLP}_{\mathcal{CS}}$. In this section, it is sufficient for $\mathcal{CS}$ to be schematic and axiomatically appropriate.

Lemma 10. *Let a term of size* $O(k2^l)$ *be defined by*

$$t_{k \to l} = \mathrm{conj}(t_k, l) \ .$$

For any binary exponential graph $G = \langle V, E \rangle$ *with* $|V| = 2^k$ *and any set* $C \subseteq V$ *of size* 2^l,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k \to l} : (F_V \to F_C) \ .$$

Proof. $|\mathrm{conj}(t_k, l)| = O\left(|t_k| 2^l\right) = O(k2^l)$.

All l-conjuncts p_i of F_C, where $i \in C$, must be k-conjuncts of F_V. Thus, for any of them by Lemma 7, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_k : (F_V \to p_i)$. Now, by Lemma 8, we have $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{conj}(t_k, l) : (F_V \to F_C)$. □

Lemma 11. *Let a term of size $O(l)$ be defined by*

$$t_{l\to\mathrm{edge}} = \mathrm{syl}(t_l, \mathrm{disj}) \ .$$

For any binary exponential graph $G = \langle V, E\rangle$, any set $C \subseteq V$ of size 2^l, and any edge $e \in E$,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{l\to\mathrm{edge}} : (F_C \to F_e) \qquad \Longleftrightarrow \qquad e \text{ is covered by } C.$$

Proof. $|\mathrm{syl}(t_l, \mathrm{disj})| = |t_l| + |\mathrm{disj}| + 5 = O(l) + O(1) = O(l)$.
($\Longleftarrow$). If $i \in e \cap C$ is the vertex in C that covers e, then p_i is a disjunct of F_e, so $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{disj} : (p_i \to F_e)$ by Lemma 9. But p_i is also an l-conjunct of F_C, so, by Lemma 7, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_l : (F_C \to p_i)$. Finally, $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{syl}(t_l, \mathrm{disj}) : (F_C \to F_e)$ by Lemma 6.

($\Longrightarrow$). If C does not cover e, it is easy to see that $F_C \to F_e$ is not valid, therefore, $\mathsf{rLP}_{\mathcal{CS}} \nvdash s : (F_C \to F_e)$ for any term s. □

Lemma 12. *Let a term of size $O(l2^m)$ be defined by*

$$s_{l\to m} = \mathrm{conj}(t_{l\to\mathrm{edge}}, m) \ .$$

For any binary exponential graph $G = \langle V, E\rangle$ with $|E| = 2^m$ and any set $C \subseteq V$ of size 2^l,

$$\mathsf{rLP}_{\mathcal{CS}} \vdash s_{l\to m} : (F_C \to F_G) \qquad \Longleftrightarrow \qquad C \text{ is a vertex cover for } G.$$

Proof. $|\mathrm{conj}(t_{l\to\mathrm{edge}}, m)| = O(|t_{l\to\mathrm{edge}}| 2^m) = O(l2^m)$.
($\Longleftarrow$). If C is a vertex cover, then $\mathsf{rLP}_{\mathcal{CS}} \vdash t_{l\to\mathrm{edge}} : (F_C \to F_e)$ for all $e \in E$, by Lemma 11. All m-conjuncts of F_G are F_e's with $e \in E$. Hence, by Lemma 8, $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{conj}(t_{l\to\mathrm{edge}}, m) : (F_C \to F_G)$.

($\Longrightarrow$). If C is not a vertex cover, by Lemma 5.3, formula $F_C \to F_G$ is not valid, hence $\mathsf{rLP}_{\mathcal{CS}} \nvdash s : (F_C \to F_G)$ for any term s. □

Theorem 13. *Let a term of size $O(k2^l) + O(l2^m)$ be defined by*

$$t_{k\to l\to m} = \mathrm{syl}(t_{k\to l}, s_{l\to m}) \ .$$

For any binary exponential graph $G = \langle V, E\rangle$ with $|V| = 2^k$ and $|E| = 2^m$ and any integer $0 \le l \le k$,

$$G \text{ has a vertex cover of size } \le 2^l \qquad \Longrightarrow \qquad \mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l\to m} : (F_V \to F_G) \ .$$

Proof. $|\mathrm{syl}(t_{k\to l}, s_{l\to m})| = |t_{k\to l}| + |s_{l\to m}| + 5 = O(k2^l) + O(l2^m)$.

By Lemma 10, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l} : (F_V \to F_C)$ for any set $C \subseteq V$ of size 2^l. If G has a vertex cover of size $\le 2^l$, it can be enlarged to a vertex cover of size 2^l. Let C be such a vertex cover of size 2^l. Then, by Lemma 12, $\mathsf{rLP}_{\mathcal{CS}} \vdash s_{l\to m} : (F_C \to F_G)$. Thus, by Lemma 6, $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{syl}(t_{k\to l}, s_{l\to m}) : (F_V \to F_G)$. □

Note that the term $t_{k\to l\to m}$ depends only on size 2^l of a vertex cover C and the numbers of vertices and edges of G.

5 Reduction from Vertex Cover, Part II

Earlier, we promised to show that the choice of a particular axiomatization for the propositional logic has no impact on our results. Indeed, for all results in Sect. 4 as well as for the $\Longleftarrow$-directions in Sect. 3, any finite schematic axiomatization would suffice. For an alternative set of propositional axiom schemes, the constants would simply be replaced by corresponding ground terms that justify the former axioms in the new system. These new terms would have size $O(1)$. It follows from the proof of Theorem 13 that the derivation of $F_V \to F_G$ we intended to represent by term $t_{k\to l\to m}$ fails. We use the condition of schematic injectivity to make sure that no other derivation of tautology $F_V \to F_G$ accidentally falls under the scope of $t_{k\to l\to m}$. In doing so, it is instrumental that we can provide a term (not necessarily a constant) that justifies all tautologies from a particular scheme and only them. Although non-atomic terms containing $+$ typically justify several schemes of formulas even if $\mathcal{CS}$ is schematically injective, it is possible to justify all propositional tautologies by $+$-free terms (see [Art01]), which justify at most one scheme. Using this observation, it is not hard to show that our results (including the ones to follow in this section) are, in fact, independent of the propositional axiom schemes chosen for A1.

To finish the polynomial-time reduction from BVC to $\mathsf{rLP}_{\mathcal{CS}}$ it now remains to prove the other direction:

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l\to m} : (F_V \to F_G) \qquad \Longrightarrow \qquad G \text{ has a vertex cover of size} \leq 2^l.$$

In this section, we again need the strongest restrictions on $\mathcal{CS}$: to be axiomatically appropriate and schematically injective.

Lemma 14 (Converse to Lemma 10)

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l} : H \qquad \Longrightarrow \qquad \begin{array}{l} H = B \to D, \\ \textit{where } D \textit{ is a balanced conjunction of depth} \geq l \\ \textit{whose all } l\textit{-conjuncts are } k\textit{-conjuncts of } B. \end{array}$$

Proof. By definition, $t_{k\to l} = \mathrm{conj}(t_k, l)$, so by Lemma 8, it justifies only implications $B \to C_1 \wedge \cdots \wedge C_{2^l}$ with $\mathsf{rLP}_{\mathcal{CS}} \vdash t_k : (B \to C_i)$ for $i = 1, \ldots, 2^l$. By Lemma 7, term t_k only justifies implications from a formula to its k-conjuncts. □

Lemma 15 (Converse to Lemma 11)

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{l\to \mathrm{edge}} : H \qquad \Longrightarrow \qquad \begin{array}{l} H = B \to D_1 \vee D_2, \\ \textit{where either } D_1 \textit{ or } D_2 \textit{ is an } l\textit{-conjunct of } B. \end{array}$$

Proof. By definition, $t_{l\to \mathrm{edge}} = \mathrm{syl}(t_l, \mathrm{disj})$. By Lemma 6, H can only be an implication $B \to D$ such that $\mathsf{rLP}_{\mathcal{CS}} \vdash t_l : (B \to C)$ and $\mathsf{rLP}_{\mathcal{CS}} \vdash \mathrm{disj} : (C \to D)$ for some C. By Lemma 9, the latter statement implies that $D = D_1 \vee D_2$ with $C = D_i$ for some $i = 1, 2$. By Lemma 7, D_i is an l-conjunct of B. □

Lemma 16 (Converse to Lemma 12)

$$\mathsf{rLP}_{\mathcal{CS}} \vdash s_{l\to m} : H \qquad \Longrightarrow \qquad \begin{array}{l} H = B \to (C_1 \vee D_1) \wedge \cdots \wedge (C_{2^m} \vee D_{2^m}), \\ \textit{where either } C_i \textit{ or } D_i \textit{ is an } l\textit{-conjunct of } B \\ \textit{for each } i = 1, \ldots, 2^m. \end{array}$$

Proof. By definition, $s_{l\to m} = \mathrm{conj}(t_{l\to\mathrm{edge}}, m)$. By Lemma 8, H must be an implication from some B to a balanced conjunction of depth $\geq m$ such that, for all its m-conjuncts F, $\mathsf{rLP}_{\mathcal{CS}} \vdash t_{l\to\mathrm{edge}} : (B \to F)$. By Lemma 15, each of these m-conjuncts must be a disjunction with one of the disjuncts being an l-conjunct of B. □

Theorem 17 (Converse to Theorem 13)

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l\to m} : H \implies \begin{array}{l} H = B \to (C_1 \vee D_1) \wedge \cdots \wedge (C_{2^m} \vee D_{2^m}), \\ \textit{and there is a size} \leq 2^l \textit{ set } X \textit{ of } k\textit{-conjuncts of } B \\ \textit{with either } C_i \in X \textit{ or } D_i \in X \textit{ for each } i = 1, \ldots, 2^m. \end{array}$$

Proof. By definition, $t_{k\to l\to m} = \mathrm{syl}(t_{k\to l}, s_{l\to m})$. By Lemma 6, $H = B \to F$ with (a) $\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l} : (B \to Q)$, (b) $\mathsf{rLP}_{\mathcal{CS}} \vdash s_{l\to m} : (Q \to F)$ for some Q. From (a), by Lemma 14, $Q = Q_1 \wedge \cdots \wedge Q_{2^l}$ whose all l-conjuncts Q_i's are also k-conjuncts of B. So $X = \{Q_i \mid i = 1, \ldots, 2^l\}$ is a size $\leq 2^l$ set (with possible repetitions) of k-conjuncts of B. It follows now from (b), by Lemma 16, that $F = (C_1 \vee D_1) \wedge \cdots \wedge (C_{2^m} \vee D_{2^m})$ with either C_i or D_i being an l-conjunct of Q for each $i = 1, \ldots, 2^m$, i.e., with either $C_i \in X$ or $D_i \in X$ for each $i = 1, \ldots, 2^m$. □

Theorem 18. *For any binary exponential graph $G = \langle V, E\rangle$ with $|V| = 2^k$ and $|E| = 2^m$ and any integer $0 \leq l \leq k$,*

$$\mathsf{rLP}_{\mathcal{CS}} \vdash t_{k\to l\to m} : (F_V \to F_G) \iff G \textit{ has a vertex cover of size} \leq 2^l.$$

Proof. The $\Longleftarrow$-direction was proven in Theorem 13. We now prove the $\Longrightarrow$-direction. $F_V \to F_G$ already has the form prescribed by Theorem 17. The only k-conjuncts of F_V are sentence letters $p_1, \ldots, p_{2^k}$. Therefore, there must exist a set X of $\leq 2^l$ of these sentence letters such that for each m-conjunct F_e of F_G at least one of its disjuncts, $p_{v_1(e)}$ or $p_{v_2(e)}$, is in X. This literally means that in G there is a set of $\leq 2^l$ vertices that covers all edges. □

Theorem 19. *For an axiomatically appropriate and schematically injective $\mathcal{CS}$, derivability in $\mathsf{rLP}_{\mathcal{CS}}$ is NP-complete.*

Proof. It was proven in [Kru03] that $\mathsf{rLP}_{\mathcal{CS}}$ is in NP. It is easy to see that both F_V and F_G have size polynomial in the size of G. As for term $t_{k\to l\to m}$, it was shown in Theorem 13 that $|t_{k\to l\to m}| = O(k2^l) + O(l2^m)$, which is polynomial in the size of G provided $l \leq k$ (BVC for $l > k$ is trivial). Thus, Theorem 18 shows that $\mathsf{rLP}_{\mathcal{CS}}$ is NP-hard. □

6 Other Justification Logics

Justification counterparts J, JD, JT, J4, and JD4 of modal logics K, D, T, K4, and D4 respectively have been developed in [Bre00] (see also [Art08a]). In addition, there are several hybrid logics combining justifications and epistemic modalities for multiple agents: $\mathsf{T}_n\mathsf{LP}_{\mathcal{CS}}$, $\mathsf{S4}_n\mathsf{LP}_{\mathcal{CS}}$, and $\mathsf{S5}_n\mathsf{LP}_{\mathcal{CS}}$ (see [Art06]). It

was shown in [Kuz08a] that their reflected fragments $\mathsf{rJ4}_{\mathcal{CS}}$, $\mathsf{rJD4}_{\mathcal{CS}}$, $\mathsf{rT}_n\mathsf{LP}_{\mathcal{CS}}$, $\mathsf{rS4}_n\mathsf{LP}_{\mathcal{CS}}$, and $\mathsf{rS5}_n\mathsf{LP}_{\mathcal{CS}}$ are axiomatized by the same $*$-calculus as $\mathsf{rLP}_{\mathcal{CS}}$, whereas axiomatization for $\mathsf{rJ}_{\mathcal{CS}}$, $\mathsf{rJD}_{\mathcal{CS}}$, and $\mathsf{rJT}_{\mathcal{CS}}$ is obtained by dropping $*$A5 for arbitrary terms while simultaneously integrating it into $*\mathcal{CS}$ for constants. This immediately yields that the Derivability Problem for all these logics is in NP for any schematic $\mathcal{CS}$ (see [Kuz08a]).

For lack of space, we cannot provide sufficient details here; we will just mention that hybrid logics $\mathsf{rT}_n\mathsf{LP}_{\mathcal{CS}}$, $\mathsf{rS4}_n\mathsf{LP}_{\mathcal{CS}}$, and $\mathsf{rS5}_n\mathsf{LP}_{\mathcal{CS}}$ are conservative over $\mathsf{rLP}_{\mathcal{CS}'}$, where $\mathcal{CS}'$ is the modality-free part of $\mathcal{CS}$. On the other hand, $\mathsf{rJ}_{\mathcal{CS}}$, $\mathsf{rJD}_{\mathcal{CS}}$, $\mathsf{rJT}_{\mathcal{CS}}$, $\mathsf{rJ4}_{\mathcal{CS}}$, and $\mathsf{rJD4}_{\mathcal{CS}}$ are strictly weaker than $\mathsf{rLP}_{\mathcal{CS}}$, but all the reasoning involved in constructing term $t_{k\to l\to m}$ can easily be performed in them too.

Theorem 20. *For an axiomatically appropriate and schematically injective constant specification $\mathcal{CS}$, the Derivability Problem for $\mathsf{rJ}_{\mathcal{CS}}$, $\mathsf{rJD}_{\mathcal{CS}}$, $\mathsf{rJT}_{\mathcal{CS}}$, $\mathsf{rJ4}_{\mathcal{CS}}$, $\mathsf{rJD4}_{\mathcal{CS}}$, $\mathsf{rT}_n\mathsf{LP}_{\mathcal{CS}}$, $\mathsf{rS4}_n\mathsf{LP}_{\mathcal{CS}}$, and $\mathsf{rS5}_n\mathsf{LP}_{\mathcal{CS}}$ is NP-complete.*

Acknowledgments

We are grateful to Sergei Artemov for playing the role of a catalyst for this research project. We thank the anonymous referees for their comments.

References

[AK06] Artemov, S.N., Kuznets, R.: Logical omniscience via proof complexity. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 135–149. Springer, Heidelberg (2006)

[Art95] Artemov, S.N.: Operational modal logic. Technical Report MSI 95–29, Cornell University (December 1995)

[Art01] Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)

[Art06] Artemov, S.N.: Justified common knowledge. Theoretical Computer Science 357(1-3), 4–22 (2006)

[Art08a] Artemov, S.N.: The logic of justification. Technical Report TR–2008010, CUNY Ph.D. Program in Computer Science (September 2008)

[Art08b] Artemov, S.N.: Why do we need Justification Logic? Technical Report TR–2008014, CUNY Ph.D. Program in Computer Science (September 2008)

[BB93] Bonet, M.L., Buss, S.R.: The deduction rule and linear and near-linear proof simulations. Journal of Symbolic Logic 58(2), 688–709 (1993)

[Bre00] Brezhnev, V.N.: On explicit counterparts of modal logics. Technical Report CFIS 2000–05, Cornell University (2000)

[Kru03] Krupski, N.V.: On the complexity of the reflected logic of proofs. Technical Report TR–2003007, CUNY Ph.D. Program in Computer Science (May 2003)

[Kru06] Krupski, N.V.: On the complexity of the reflected logic of proofs. Theoretical Computer Science 357(1-3), 136–142 (2006)

[Kuz00] Kuznets, R.: On the complexity of explicit modal logics. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 371–383. Springer, Heidelberg (2000); Errata concerning the explicit counterparts of $\mathcal{D}$ and $\mathcal{D}4$ are published as [Kuz08b]

[Kuz08a] Kuznets, R.: Complexity Issues in Justification Logic. PhD thesis, CUNY Graduate Center (May 2008)

[Kuz08b] Kuznets, R.: Complexity through tableaux in justification logic. In: Abstracts of Plenary Talks, Tutorials, Special Sessions, Contributed Talks of Logic Colloquium (LC 2008), Bern, Switzerland, pp. 38–39 (Abstract) (July 3-8, 2008)

[Kuz08c] 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)

[Lad77] Ladner, R.E.: The computational complexity of provability in systems of modal propositional logic. SIAM Journal on Computing 6(3), 467–480 (1977)

[Mil07] Milnikel, R.: 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)

Taming Modal Impredicativity: Superlazy Reduction

Ugo Dal Lago[1], Luca Roversi[2], and Luca Vercelli[3]

[1] Dipartimento di Informatica, Università di Bologna
http://www.cs.unibo.it/~dallago/
[2] Dipartimento di Informatica, Università di Torino
http://www.di.unito.it/~rover/
[3] Dipartimento di Matematica, Università di Torino
http://www.di.unito.it/~vercelli/

Abstract. Pure, or type-free, Linear Logic proof nets are Turing complete once cut-elimination is considered as computation. We introduce *modal impredicativity* as a new form of impredicativity causing cut-elimination to be problematic from a complexity point of view. Modal impredicativity occurs when, during reduction, the conclusion of a residual of a box b interacts with a node that belongs to the proof net *inside* another residual of b. Technically speaking, *superlazy reduction* is a new notion of reduction that allows to control modal impredicativity. More specifically, superlazy reduction replicates a box only when all its copies are opened. This makes the overall cost of reducing a proof net finite and predictable. Specifically, superlazy reduction applied to any pure proof nets takes primitive recursive time. Moreover, any primitive recursive function can be computed by a pure proof net via superlazy reduction.

Keywords: Linear logic, implicit computational complexity, proof theory.

1 Introduction

Predicativity is a logical concept known from a century, starting from Russel's work. It has various technical meanings. All of them, however, refer implicitly or explicitly to some form of aciclicity (see [3] for an excellent survey). Impredicative definitions or logical rules in a logical system may lead to logical paradoxes. On the other hand, if logical systems are interpreted as programming languages (via the Curry-Howard correspondence), impredicativity may lead to type systems and programming languages with high expressive power.

In this paper, we introduce the notion *modal impredicativity*. We start from Linear Logic which gives first-order status to structural rules (on the logical side) and to duplication and erasure (on the computational side). The very definition of modal impredicativity refers to *boxes*, i.e., those portions of proof nets, related to the modal meaning of formulae, that may be duplicated. During cut-elimination, a duplication occurs when a box interacts with a contraction node,

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 137–151, 2009.

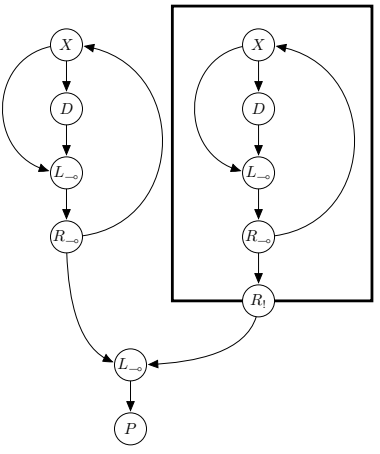


Fig. 1. The pure proof net $\Delta\Delta$

which corresponds to an instance of the structural rule contraction in a logical derivation. Boxes allow to structure proofs into *layers*: any rule instance has a *level*, the number of boxes into which it is contained. Focusing our attention to boxes is the reason why our notion of impredicativity is dubbed as modal. Specifically, modal impredicativity occurs when, during reduction, the *root* of a residual of a box b interacts with a node that belongs to the residual of the proof net lying *inside* b. The paradigmatic example where an interaction of this kind occurs is in the (pure) proof net in Figure 1, which encodes the prototypical *non normalizing* lambda-term $(\lambda x.xx)(\lambda x.xx)$. Call b the (unique) box in Figure 1. After two reduction steps we get two copies b' and b'' of b. Two further reduction steps plug the root of b' as premise of the node X belonging to the second copy b'' of b. This is a basic form of one-step-long cycle, since the content of b interacts with the root of b itself. Compared to what happens classically, *self-copying* plays the rôle of *self-application* or *self-definition*. Notice that the cycles we are speaking about can have length greater than one. As an example, consider the proof net corresponding to the lambda term $(\lambda x.\lambda y.xyx)(\lambda x.\lambda y.xyx)(\lambda x.\lambda y.xyx)$: it includes two boxes b_1 and b_2 where b_1 copies b_2 and b_2 copies b_1.

Our long-term goal is to define proper restrictions on Linear Logic allowing to control modal impredicativity. This paper is just the first step towards this goal. What we define here is a new notion of *reduction* for Linear Logic proof nets which rules out the previously described cyclic phenomenon dynamically, i.e. at the level of the graph-theoretic rewriting relation which governs proof net reduction. This way, we break impredicative cycles, while keeping the freedom of *statically* compose pure proof nets.

Light Logics and Modal Impredicativity. We now recall how the known subsystems of Linear Logic, introduced as characterizations of certain complexity classes, work. Let us call them light logics, for short. Proof-theoretically, they parsimoniously use the contraction rule. On the computational side, they control

the duplication of structure. Technically, we currently know two ways of controlling the use of duplication. One is *stratification*. The other one is what we like to call *boundedness*.

Stratification is a structural constraint that, at the dynamic level, has the following meaning: one reduction step at level n can only increase the complexity of the underlying proof at levels (strictly) higher than n. This is achieved by dropping dereliction and digging as logical rules. The consequence is the control over the dimension of every single reduct, that reflects on the overall control of reduction time. The mechanism is implicit in the structural and combinatorial properties of proofs and is totally independent from its logical soundness. In stratified systems, the level of any node *cannot* change during reduction. As a consequence, any stratified system, by definition, cannot be modally impredicative because the nodes inside a box b cannot interact with the root of any copy of b. Elementary Linear Logic, Light Linear Logic [6] and their affine versions use *stratification*.

Concerning boundedness, recall that in ordinary Linear Logic, $!A$ is semantically equivalent to $A^* = \bigcup_{n\in\mathbf{N}} \underbrace{(A \otimes \ldots \otimes A)}_{n}$. *Boundedness* refers to various methodologies that, informally, put $!A$ in correspondence to a *finite* subset of A^*. Computationally, this means the number of copies of each box in a proof can somehow be statically or dynamically predicted, i.e. bounded. This way, we automatically get a system that cannot be modally impredicative, since in bounded systems $!A$ cannot be equal to $!A \otimes A$ (and this principle seems necessary to have self-copying). Soft Linear Logic [8] and Bounded Linear Logic [4] use *boundedness*.

Superlazy Reduction and Modal Impredicativity. Superlazy reduction is a new notion of reduction for Linear Logic (pure) proof nets. It is specifically designed to control modal impredicativity. Under superlazy reduction, any box b can interact with a tree of contraction, dereliction, digging and weakening nodes only if the global result of this interaction somehow reduces the overall complexity of the proof net, namely when it produces (possibly many) "open" copies of b. If this is not the case, reduction is blocked and cannot be performed. This way modal impredicativity is automatically ruled out, since whenever the *content* of b is copied, b *as a box* is destroyed and no residuals of b are produced. Technically, this is ensured by prescribing that reduction can happen only when the box is faced with a *derelicting tree of nodes*, a key notion introduced in Section 3.

Superlazy Reduction and Primitive Recursion. The calculus we obtain by adopting superlazy reduction over pure proof nets is still powerful enough to characterize the class PR of primitive recursive functions. We show the characterization under a standard pattern. As for *soundness*, we prove that every pure proof net G can be rewritten to its normal form in time bounded by a primitive recursive function in the size of G. This is remarkable by itself, since the mere fact that superlazy reduction *computes something* is interesting by itself, considered the strong

requirements superlazy reduction must satisfy. As for *completeness*, every function in PR can be represented as a pure proof net, even under superlazy reduction.

Superlazy Reduction and Expressive Power. We here want to make some observations about pure proof nets and superlazy reduction as a paradigmatic programming language. The set of terms we can program with are pure proof nets coming from Linear Logic. Namely, we can use every lambda-term as a program, $\Delta\Delta$ (Figure 1) included. However, we do not have the standard unconstrained reduction steps, which, by simulating the usual beta-reduction, allow to embed pure, i.e. untyped, lambda-calculus into pure proof nets. In particular, the proof net $\Delta\Delta$ is normal in our setting, the reason being that we can never reach a situation where the "amount" of open copies of the box it contains is known in advance.

Related Work. Several authors have used some notion of predicativity as a way to restrict the expressive power of programming languages or logical systems. We here recall some of them, without any hope of being exhaustive. In [11], as a first example, Danner and Leivant presented a variant of second order λ-calculus obtained imposing a restriction on second order quantification. Such a restriction has semantic flavor: all the *types* have a *rank*, an ordinal number and universal quantification can only be instantiated if the witness has a proper rank. This way they get a characterization of primitive recursion. Another, earlier, example is Leivant's predicative recursion [9]: if predicativity is imposed on ordinary primitive recursion (on any word algebra), one gets a characterization of polynomial time computable functions. Further work shows how other classes can be characterized with similar tools [10,12]. A final example is Simmons' fine analysis of tiering [14]. All the cited proposals, however, share a property which make them fundamentally different from superlazy reduction: predicativity is enforced through static constraints, i.e., constraints on programs rather than on the underlying reduction relation. The first author has recently proposed a characterization of primitive recursion by a fragment of Gödel's T [1].

On the other hand, restricted notions of reduction on Linear Logic proof nets have appeared in the literature. This includes, for example, Girard's closed reduction [5] or head linear reduction [13]. None of them, however, decreases the expressive power of the logical systems on top of which they are applied, as superlazy reduction does.

Paper Outline. Section 2 recalls pure, i.e., untyped, proof nets and defines dereliction trees and superlazy reduction. Section 3 proves the primitive recursive soundness of superlazy reduction on pure proof nets. Section 4 shows that even under superlazy reduction, pure proof nets remain expressive enough to represent all the primitive recursive functions. Section 5 presents some further developments on the ideas presented in the paper. An extended version of this paper including all proofs is available [2].

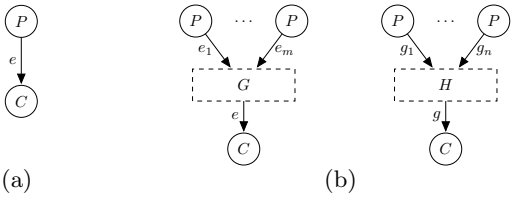


Fig. 2. Base cases

2 Pure Proof Nets

Pure proof nets are graph-like structures corresponding to proofs. The set of labels for the nodes is $\{P, C, W, X, R_{\multimap}, L_{\multimap}, R_{\otimes}, L_{\otimes}, R_{!}, L_{!}, D, N\}$. All nodes, but P and C, correspond to the usual proof net labels. We use P and C for the sake of uniformity, getting graphs without dangling edges. Figure 2(a) says that a wire is a proof net. Given the two proof nets in 2(b) we can build those in Figure 3. The inductive rule at the end of Figure 2(a) introduces (modal) *boxes*.

Please notice that the proof nets introduced here are slightly different from the usual ones. In particular, there is not any explicit node playing the rôle of the cut rule or of axioms. Moreover, proof net conclusions are partitioned into one proper conclusion and some premises. This way, proof nets get an intuitionistic flavor which makes the correspondence with lambda-terms more evident.

Reduction Rules. The reduction rules for the proof nets are the usual ones. We omit the obvious linear rules $\longrightarrow_{\multimap}, \longrightarrow_{\otimes}$, and we just recall the *modal* reduction rules in Figure 4. Call $\longrightarrow$ the contextual closure of the rewriting steps $\longrightarrow_{\multimap}, \longrightarrow_{\otimes}, \longrightarrow_{D}, \longrightarrow_{X}, \longrightarrow_{W}, \longrightarrow_{N}, \longrightarrow_{M}$. The reflexive and transitive closure of $\longrightarrow$ is $\longrightarrow^{*}$.

The reduction rules $\longrightarrow_{X}$ and $\longrightarrow_{N}$ are the only ones somehow increasing the size of the underlying proof-net: the first one copies a box, while the second one puts a box inside another box. Superlazy reduction, as we will see shortly, does not simply eliminate those rules, but rather forces them to be applicable only in certain contexts, i.e., only when those rules are part of a sequence of modal rewriting rules which have a globally predictable behavior.

2.1 Superlazy Reduction

We shall be able to prove a soundness result about the cost of the reduction of the proof nets relatively to a *superlazy* version of $\longrightarrow$ that requires the notion of *derelicting tree*.

Derelicting trees. For every proof net G, let us assume that the *cost* of traversing any X-node of G is 0, any D-node is -1, any W-node is 0 and any N-node is $+1$. The cost of a path from node u to node v is the sum of the costs of nodes in the path *including u and v*. A *derelicting tree* in G is a subgraph t of G that satisfies the following four conditions:

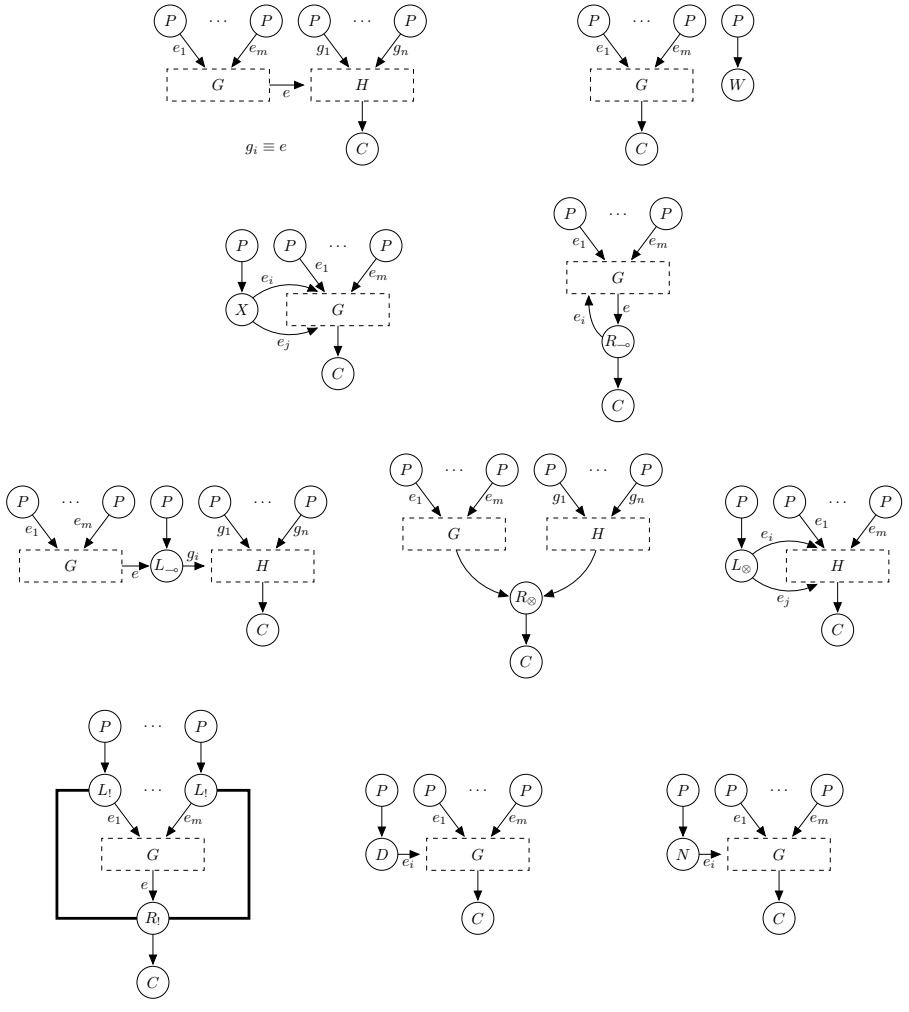


Fig. 3. Inductive cases

1. t only contains nodes X, D, N, W; so it must be a tree, and we call w its root;
2. the leaves of t are labelled either with D or with W;
3. for every leaf v labelled with D in t, the cost of the path from w to v in t is -1;
4. the cost of any other path in t starting from w is nonnegative.

Figure 5(a) shows an example of a dereliction tree. Conditions 1. and 2. are trivially satisfied. The cost of $v_1v_2v_4v_7v_{10}$ is -1 and the same for $v_1v_3v_6$. Finally, any other path starting from v_1 has nonnegative cost. For example, $v_1v_3v_5v_8$ has cost 1.

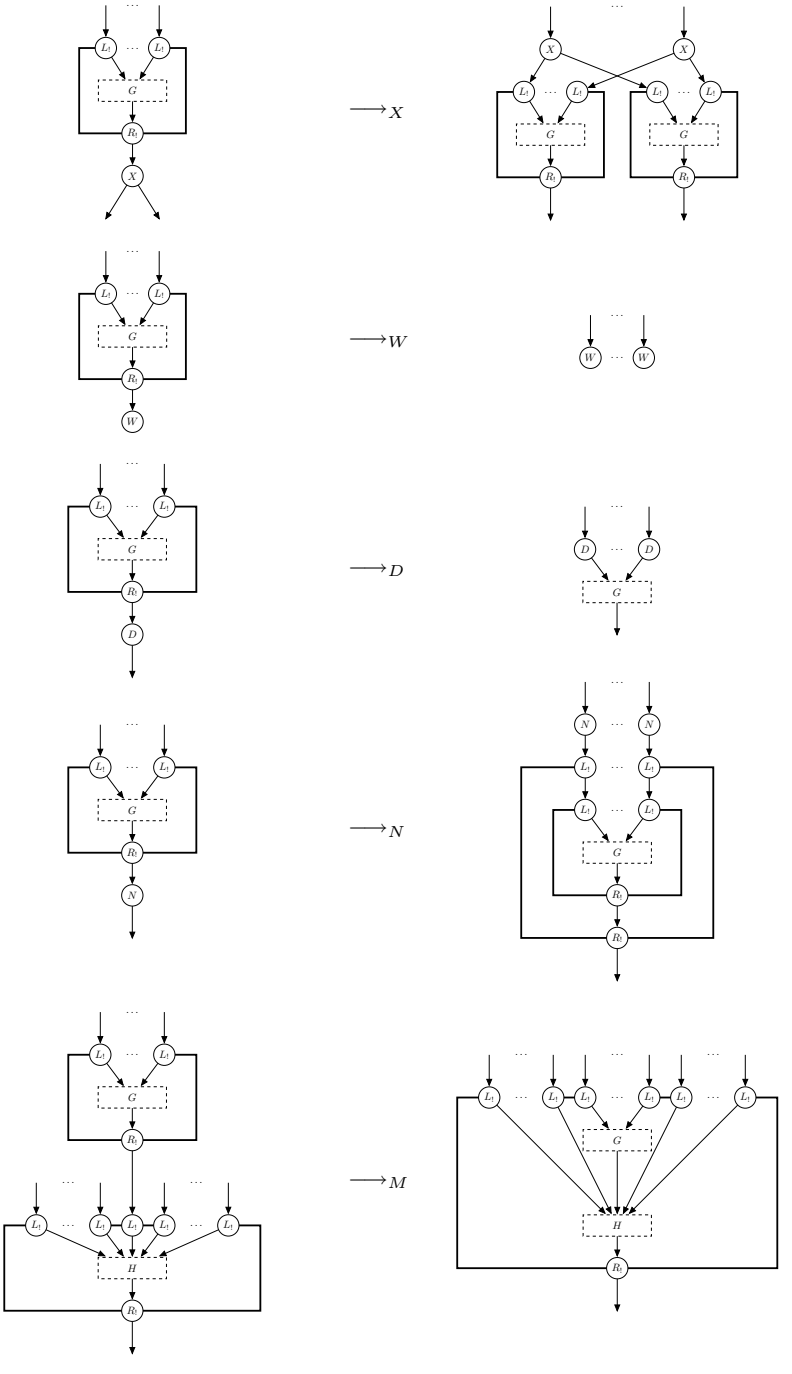


Fig. 4. Modal graph rewriting rules

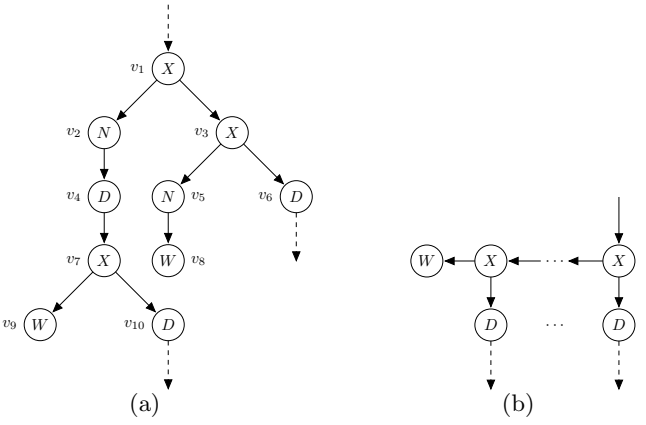


Fig. 5. A generic derelicting tree (a) and a bounded spine (b)

Figure 5(b) depicts a remarkable instance of derelicting tree: an *n-bounded spine* with n occurrences of X nodes that we shall represent as a dashed box with name nX.

Superlazy Reduction Step and Rewriting System on Pure Proof Nets. The *superlazy normalization step* is $\rightarrow_{XNDW}$, defined in Figure 6, ∇t being any derelicting tree. $\rightsquigarrow_{-\circ}, \rightsquigarrow_{\otimes}, \rightsquigarrow_{M}, \rightsquigarrow_{XNDW}$ denote the *surface* contextual closure of the rewriting steps $\longrightarrow_{-\circ}, \longrightarrow_{\otimes}, \longrightarrow_{M}, \longrightarrow_{XNDW}$. "Surface" means that we never apply a reduction inside a box. $\rightsquigarrow$ is the union of $\rightsquigarrow_{-\circ}, \rightsquigarrow_{\otimes}, \rightsquigarrow_{M}, \rightsquigarrow_{XNDW}$. The reflexive and transitive closure of $\rightsquigarrow$ is $\rightsquigarrow^*$.

Superlazy reduction is a *very* restricted notion of reduction. In particular, it is almost useless when applied to proof nets obtained from ordinary lambda-terms via the usual, uniform encoding. In particular:

- If terms are encoded via the so-called call-by-name encoding (i.e., the one induced by Girard's correspondence $(A \rightarrow B)^{\circ} \equiv !A^{\circ} \multimap B^{\circ}$), then any redex

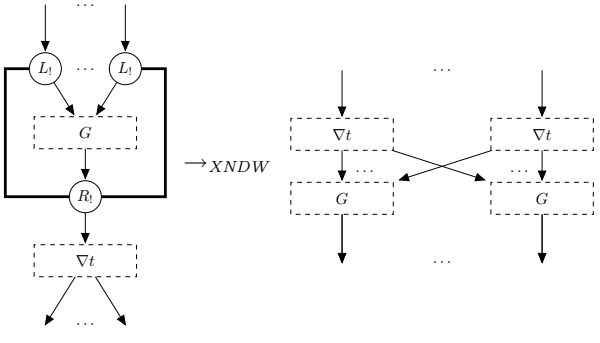


Fig. 6. Superlazy cut elimination step

$(\lambda x.M)N$ where M consists of an application LP and x appears free in P, cannot be reduced in the corresponding proof net: a box would be faced with something different from a derelicting tree.
- On the other hand, if terms are encoded via the call-by-value encoding (i.e. via the correspondence $(A \to B)^\circ \equiv !(A^\circ \multimap B^\circ)$), then any redex $(\lambda x.M)N$ where M consists itself of an abstraction $\lambda y.L$ and x appears free in L cannot be reduced in the corresponding proof net.

Unfortunately, we do not even know any criteria allowing to guarantee that certain proof nets can be reduced to normal form (w.r.t. ordinary reduction) by way of superlazy reduction. Moreover, there currently isn't any result characterizing the class of normal forms w.r.t. superlazy reduction; this is in contrast, for example, to lambda calculus and call-by-value reduction, where the cbv normal form of any (closed) term M (if any) is always a value. This is why proving that proof nets are complete w.r.t. some given class of functions, under the superlazy reduction, is non-trivial. Nonetheless, we explicitly prove the completeness of superlazy reduction (see Section 4). The next section shows why both $(A \to B)^\circ \equiv !A^\circ \multimap B^\circ$ and $(A \to B)^\circ \equiv !(A^\circ \multimap B^\circ)$ do not work well here: superlazy reduction of any proof net always terminates in a time bounded by suitable primitive recursive functions.

Linear Logic with superlazy reduction can be seen as a generalization of the principles of Soft Linear Logic. Every time a box is replicated, that box is opened. According to this vision, a derelicting tree with m leaves is similar to a multiplexor node of rank m. However, the structure of SLL gives a further restriction: if k is the rank of the proof net G, that is the maximum rank of the multiplexers in G, we are sure that every box of G will be copied at most k times. Such a restriction leads to a polytime bound (see [8]).

3 Soundness

We prove the result starting with a restriction $\hookrightarrow$ on the relation $\rightsquigarrow$. $\hookrightarrow$ is simply the union of $\rightsquigarrow_{-\circ}$, $\rightsquigarrow_{\otimes}$, $\rightsquigarrow_M$ and $\rightsquigarrow_{XNDW}$, where any $\rightsquigarrow_M$ can only be applied to nets which only contain $\rightsquigarrow_M$ redexes. In other words, $\rightsquigarrow_M$ is postponed as much as possible. This makes our arguments simpler without loss of generality, as we now show. $\hookrightarrow^k$ will denote a reduction with k steps of $\hookrightarrow$. Given a proof G and any reduction relation $\to$, $[G]_\to$ and $||G||_\to$ denote the maximum length of a reduction sequence starting in G (under $\to$) and the maximum size of any reduct of G (under $\to$), respectively. $|G|$ is the size of the proof net G.

Lemma 1. *For every proof G, $[G]_\rightsquigarrow = [G]_\hookrightarrow$ and $||G||_\rightsquigarrow = ||G||_\hookrightarrow$.*

Proof. Whenever $G \rightsquigarrow_M F \rightsquigarrow_x H$ and $x \neq M$, there are $F_1, \ldots, F_n$ (where $n \geq 1$) such that $G \rightsquigarrow_{x_1} F_1 \rightsquigarrow_{x_2} \cdots \rightsquigarrow_{x_n} F_n \rightsquigarrow_{x_{n+1}} H$, and $x_{i+1} = M$ whenever $x_i = M$. For example, if $G \rightsquigarrow_M F \rightsquigarrow_{XNDW} H$ and the box copied in the second step is exactly the one created by the first step, then clearly

$$G \rightsquigarrow_{XNDW} J \rightsquigarrow_{XNDW} H.$$

As a consequence, for any sequence $G_1 \rightsquigarrow \cdots \rightsquigarrow G_n$ there is another sequence $F_1 \hookrightarrow \cdots \hookrightarrow F_m$ such that $G_1 = F_1$, $G_n = F_m$ and $m \geq n$. This proves the first claim. Now, observe that for any $1 \leq i \leq n$ there is j such that $|F_j| \geq |G_i|$: a simple case analysis suffices. This concludes the proof. □

The following definition is the main ingredient since it allows to structure the proof of soundness inductively. A reduction sequence $\sigma \equiv G \rightsquigarrow_{x_1} F \rightsquigarrow_{x_2} \ldots \rightsquigarrow_{x_k} H$ is said to be a (n, d)-*box reduction* when $x_i \neq M$ for every $1 \leq i \leq k$ and there are $r \leq n$ boxes $b_1, \ldots, b_r$ between those at level 0 in G such that the following conditions hold (let $J_1, \ldots, J_r$ be the proof nets inside $b_1, \ldots, b_r$, respectively):

- $\partial(J_1), \ldots, \partial(J_r) < d$.
- The box involved in any step $\rightsquigarrow_{XNDW}$ of σ is either a residual of $b_1, \ldots, b_r$ or a residual of a box appearing in one of $J_1, \ldots, J_r$.

Remark 1. A few observations about the above definition:

1. G may contain more than n boxes at level 0, or it may contain boxes whose depth is greater than d;
2. H is *not* necessarily a normal form. It may contain boxes at level d, or higher;
3. By definition, any node inside the boxes $b_1, \ldots, b_r$ *must have* depth at least 1. This means that if σ is a $(n, 0)$-*box reduction* it is, in fact, talking about boxes that contain proof nets with negative depth. They cannot exist, so σ only uses the linear rewriting steps $\longrightarrow_{\multimap}, \longrightarrow_{\otimes}$;
4. By definition, *every* reduction starting at any net G is a $(|G|, \partial(G))$-box reduction because $|G|$ necessarily bounds the number n of boxes, and $\partial(G)$ bounds the value d;
5. The definition is a good one *only* because we are assuming to work without the rule $\longrightarrow_M$. Otherwise (residuals of) boxes coming from $b_1, \ldots, b_r$ could be "merged" with (residuals of) boxes in G but not in the list.

Now we define a family of functions $\{f_d : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}_{d \in \mathbb{N}}$ such that every $f_d(n, m)$ bounds *both* the reduction cost *and* the size of the reducts, of every net G when performing a (n, d)-box reduction on it:

$$
\begin{aligned}
f_0(n, m) &= m \\
f_{d+1}(0, m) &= m \\
f_{d+1}(n+1, m) &= 1 + f_{d+1}(n, m) + f_d((2m+1)f_{d+1}(n, m), (2m+1)f_{d+1}(n, m))
\end{aligned}
$$

By definition, all the functions f_d are primitive recursive.

Lemma 2. *For every $d, n, m \in \mathbb{N}$, $f_{d+1}(n, m) \geq f_d(n, m)$ and $f_d(n+1, m) \geq f_d(n, m)$.*

The proof is by induction on n. We can prove the following:

Proposition 1. *Let G be any proof net and let $G \hookrightarrow^k H$ be a (n, d)-box reduction sequence with k steps. Then $k, |H| \leq f_d(n, |G|)$.*

Proof. We write $m = |G|$. We can proceed by induction on d:

- If $d = 0$, thanks to Remark 1.3 above, the reduction $G \hookrightarrow^k H$ only involves multiplicative reduction steps. They strictly reduce the size of the underlying net. So, $|H|, k \leq m = f_0(n, m)$.
- Suppose the thesis holds for d and suppose $G \hookrightarrow^k H$ is a $(n, d+1)$-box reduction. We proceed by another induction, this time on n:
  - If $n = 0$, then none of the boxes at level $d+1$ can be reduced. As a consequence, $G \hookrightarrow^k H$ only involves multiplicative reduction steps and, again,
$$|H|, k \leq m = f_{d+1}(0, m).$$
  - Suppose the thesis holds for n and suppose $G \hookrightarrow^k H$ is a $n+1$-box reduction sequence at level $d+1$. By definition, there are $r \leq n+1$ boxes $b_1 \ldots, b_r$ in G satisfying the definition above. There is clearly one index t, where $1 \leq t \leq r$ such that b_t is the last box being copied (among $b_1, \ldots, b_r$) in the sequence $G \hookrightarrow^k H$. Up to the point where b_t is copied, the reduction sequence can be considered as a $(n, d+1)$-box reduction sequence: the witness list of box is exactly $b_1, \ldots, b_{t-1}, b_{t+1}, \ldots, b_r$. After b_t is copied, on the other hand, the reduction sequence under consideration can be considered as a $(|J|, d)$-box reduction sequence, since all the boxes that will be copied are part of the residual of the content of boxes in the list $b_1, \ldots, b_r$. The reduction under consideration can be decomposed as follows
$$G \hookrightarrow^i F \rightsquigarrow_{XNDW} J \hookrightarrow^j H$$
where $G \hookrightarrow^i F$ is an $(n, d+1)$-box reduction sequence, the step $F \rightsquigarrow_{XNDW} J$ involves *exactly* b_t and $J \hookrightarrow^j H$ is a $(|J|, d)$-box reduction. Now, observe that $|b_t| \leq m$. In $|J|$ we will find at most $|F|$ copies of b_t and at most $|b_t|$ copies of the underlying derelicting tree (which can itself contain at most $|F|$ nodes). Applying both inductive hypothesis, we get
$$\begin{aligned} i, |F| &\leq f_{d+1}(n, m) \\ |J| &\leq 2m f_{d+1}(n, m) + |F| \\ j, |H| &\leq f_d(|J|, |J|) \end{aligned}$$
As a consequence:
$$\begin{aligned} k = i + 1 + j &\leq f_{d+1}(n, m) + 1 + f_d(|J|, |J|) \\ &\leq f_{d+1}(n, m) + 1 + f_d((2m+1)f_{d+1}(n, m), (2m+1)f_{d+1}(n, m)) \\ |H| \leq f_d(|J|, |J|) &\leq f_{d+1}(n, m) + 1 + f_d(|J|, |J|) \\ &\leq f_{d+1}(n, m) + 1 + f_d((2m+1)f_{d+1}(n, m), (2m+1)f_{d+1}(n, m)) \end{aligned}$$

This concludes the proof. □

Corollary 1 (Soundness). *For every $n \in \mathbb{N}$ there is a primitive recursive function $g_n : \mathbb{N} \to \mathbb{N}$ such that for every proof net G, $[G]_\rightsquigarrow, ||G||_\rightsquigarrow \leq g_{\partial(G)}(|G|)$.*

Proof. By Lemma 1, we can bound $[G]_{\hookrightarrow}$ and $||G||_{\hookrightarrow}$ rather than $[G]_{\rightsquigarrow}$ and $||G||_{\rightsquigarrow}$. Now, observe that any reduction $G \hookrightarrow^k H$ is a $(|G|, \partial(G))$-box reduction, followed by at most a linear number of M-reduction steps, which anyway shrinks the size of the underlying proof net. The thesis follows from Proposition 1. □

4 Completeness

The goal is to show the existence of an embedding of any primitive recursive function f into a pure proof net G_f such that G_f simulates f via superlazy reduction. The existence of such an embedding is what we mean by *completeness*.

Our comments at the end of Section 2 should have convinced the reader about the impossibility of proving completeness by the usual encoding of the untyped lambda-calculus into pure proof nets. We really need to tailor the encoding of data and programs in such a way that superlazy reduction *works* on them, i.e., we want to be sure that a program applied to an argument *reduces* to the intended result.

The details of the completeness proof do not fit here, but they can be found in [2], where we also also develop a slightly more general proof of soundness. Only the major ingredients towards completeness are reported here.

The first ingredient is the representation of natural numbers. Figure 7 introduces the closed proof nets $\overline{0}$ and $\overline{n}$, with $n \geq 1$. A basic observation to prove the completeness is that every $\overline{n}$ contains a n-bounded spine nX (see Fig. 5(b)). This implies that whenever $\overline{n}$ is applied to a box containing a proof net G, superlazy reduction will copy the box n times and n copies of G will appear, one applied to the next one. Computing the successor of a natural number can be done by a proof net, quite similarly to what happens in ordinary lambda calculus. Now, at least, the key notion can be given. Let G be a proof net with a single premise and a single conclusion. We write $G \smile \langle \overline{n_1}, \ldots, \overline{n_m} \rangle$ to mean the closed proof net with a single conclusion obtained by plugging the conclusion of the net $\langle \overline{n_1}, \ldots, \overline{n_m} \rangle$ (obtained by "tensoring" together $\overline{n_1}, \ldots, \overline{n_m}$) into the unique assumption of G. For every function $f : \mathbb{N}^m \to \mathbb{N}$, with arity $m \geq 0$, we

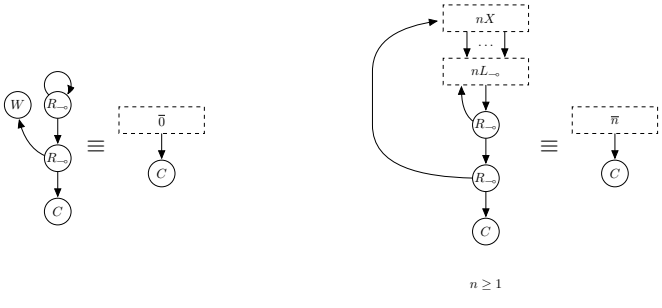


Fig. 7. Church numerals as proof nets

shall say that a proof net G_f with one premise *represents* f iff $G_f \frown \langle \overline{n_1}, \ldots, \overline{n_m} \rangle$ *superlazily* reduces to $\overline{f(n_1, \ldots, n_m)}$, for every $n_1, \ldots, n_m \in \mathbb{N}$.

The second ingredient is the possibility of freely duplicate data, i.e., natural numbers. This is possible even if the natural number being copied (or erased) does not lie inside a box. Copying a natural number n involves applying a (boxed) pair of successors and a pair of $\overline{0}$ to the net $\overline{n}$. Erasing n, on the other hand, can be performed by applying a boxed identity and another boxed identity to $\overline{n}$: this way $\overline{n}$ can be reduced itself to a boxed identity, which can be erased by cutting it against a W node.

The first two ingredients allow to encode basic primitive recursive functions and composition. To get the most important construction, namely primitive recursion itself, a third ingredient is necessary, namely iteration. Iterating n times a given function f, where $\overline{n}$ is a parameter, can be done by way of superlazy reduction: putting the proof net representing f inside a box and apply the box to $\overline{n}$ suffices. However, if the proof net representing f has some premises, the resulting proof net would only accept boxed natural numbers as arguments, and this would break the scheme which makes superlazy reduction works. The solution consists in iterating only closed functions, exploiting the higher-order nature of proof nets. The usual primitive recursion scheme can be finally obtained by using the standard technique of simulating recursion by iteration.

Theorem 1 (Completeness). *Every f in PR is represented by a proof net G_f.*

5 Further Developments

As we explained in the Introduction, this paper is just the first step in a long-term study about how to control modal impredicativity.

The are at least two distinct research directions the authors are following at the time of writing. We recall the obtained results here, pointing to further work for additional details and proofs.

First of all, the way we proved completeness of proof net reduction w.r.t. primitive recursion suggests a way of capturing Hofmann's non-size increasing computation [7] by way of a proper, further restriction to superlazy reduction. We basically need two constraints:

- Boxes can only be copied when they are closed, i.e., when they have no premises.
- Any box b can only be copied if the proof net contained in b contains at most *one* node X at level 0.

The obtained reduction relation is said to be the *non-size increasing superlazy reduction* and is denoted with $\Rightarrow$. With these constraints, non-size increasing polytime computation can be simulated by pure proof nets. Moreover:

Theorem 2. *For every $n \in \mathbb{N}$ there is a polynomial $p_n : \mathbb{N} \to \mathbb{N}$ such that for every proof net Π, $[G]_\Rightarrow, ||G||_\Rightarrow \leq p_{\partial(G)}(|G|)$*

The second direction we are considering concerns methods to keep modal impredicativity under control by more traditional static methods in the spirit of light logics. Consider the equivalence between $!A$ and $!A \otimes A$. As already pointed out, this equivalence is somehow necessary to get modal impredicativity. So, controlling it means controlling modal impredicativity. Now, suppose that the above equivalence holds *but* A only contains instances of the ! operator which are *intrinsically different* from the top-level one in $!A$. Morally, this would imply that even if a contraction node is "in" A, it cannot communicate with the box in $!A$, because they are of a different nature. This way modal impredicativity would be under control. But the question is: how to distinguish different instances of ! from each other? One (naive) answer is the following: consider a generalization of (multiplicative and exponential) Linear Logic where syntactically different copies $!_{a_1}, !_{a_2}, \ldots$ of the modal operator ! are present. As an example, take the set $\{!_n\}_{n \in \mathbb{N}}$. Then, impose the following constraint: $!_a A$ is a *legal* formula only if the operators $!_{b_1}, \ldots, !_{b_n}$ appearing in A are all different from a, i.e., if $a \notin \{b_1, \ldots, b_n\}$. We strongly believe that this way a system enjoying properties similar to those of predicative recurrence schemes [9] can be obtained.

6 Conclusions

We described modal impredicativity and a concrete tool — superlazy reduction — that controls it. Superlazy reduction on pure proof nets greatly influences the expressive power of the underlying computational model: from a Turing complete model we go down to first-order primitive recursive functions.

In a sentence, we learnt that the expressive power of a programming language system can be controlled by acting on the dynamics (i.e., the underlying reduction relation) without touching the statics (i.e., the language into which programs are written). To get the complete picture, however, we still need tools to predict which set of programs will be useful from a computational point of view.

This could have potential applications in the field of implicit computational complexity, where one aims at designing programming languages and logical systems corresponding to complexity classes. Indeed, the impact of ICC in applications crucially depends on the intensional expressivity of the proposed systems: one should be able to write programs naturally.

References

1. Dal Lago, U.: The geometry of linear higher-order recursion. In: Proc. 20th Annual Symposium on Logic in Computer Science, pp. 366–375. IEEE Computer Society, Los Alamitos (2005)
2. Dal Lago, U., Roversi, L., Vercelli, L.: Taming modal impredicativity: Superlazy reduction. Extended Version (2008), `http://arxiv.org/abs/0810.2891`
3. Feferman, S.: Predicativity. In: Shapiro, S. (ed.) The Oxford Handbook of the Philosophy of Mathematics and Logic, pp. 590–624. Oxford University Press, Oxford (2005)

4. Girard, J.-Y., Scedrov, A., Scott, P.: Bounded linear logic: A modular approach to polynomial time computability. Theor. Comput. Sci. 97, 1–66 (1992)
5. Girard, J.-Y.: Geometry of interaction 1: interpretation of system F. In: Proc. Logic Colloquium 1988, pp. 221–260 (1989)
6. Girard, J.-Y.: Light linear logic. Information and Computation 143(2), 175–204 (1998)
7. Hofmann, M.: Linear types and non-size-increasing polynomial time computation. In: Proceedings of the 14th Symposium on Logic in Computer Science, pp. 464–473. IEEE Computer Society, Los Alamitos (1999)
8. Lafont, Y.: Soft linear logic and polynomial time. Theor. Comput. Sci. 318, 163–180 (2004)
9. Leivant, D.: Stratified functional programs and computational complexity. In: Proceedings of 20th ACM Symposium on Principles of Programming Languages, pp. 325–333 (1993)
10. Leivant, D.: Ramified recurrence and computational complexity III: Higher type recurrence and elementary complexity. Ann. Pure Appl. Logic 96(1-3), 209–229 (1999)
11. Leivant, D., Danner, N.: Stratified polymorphism and primitive recursion. Math. Struct. in Comp. Science 9, 507–522 (1999)
12. Leivant, D., Marion, J.-Y.: A characterization of alternating log time by ramified recurrence. Theor. Comput. Sci. 236(1-2), 193–208 (2000)
13. Mascari, G., Pedicini, M.: Head linear reduction and pure proof net extraction. Theor. Comput. Sci. 135(1), 111–137 (1994)
14. Simmons, H.: Tiering as a recursion technique. Bulletin of Symbolic Logic 11(3), 321–350 (2005)

Positive Fork Graph Calculus*

Renata de Freitas[1], Sheila R.M. Veloso[2],
Paulo A.S. Veloso[3], and Petrucio Viana[1]

[1] Institute of Mathematics, UFF: Universidade Federal Fluminense; Niterói, Brazil
petrucio@cos.ufrj.br
[2] Systems and Computer Engin. Dept., Faculty of Engineering, UERJ: Universidade do Estado do Rio de Janeiro; Rio de Janeiro, Brazil
[3] Systems and Computer Engin. Program, COPPE, UFRJ; Universidade Federal do Rio de Janeiro; Rio de Janeiro, Brazil

Abstract. We introduce and illustrate a graph calculus for proving and deciding the positive identities and inclusions of fork algebras, i.e., those without occurrences of complementation. We show that this graph calculus is sound, complete and decidable. Moreover, the playful nature of this calculus renders it much more intuitive than its equational counterpart.

Keywords: Positive relational calculi, fork algebras, graph calculus, completeness, decidability.

1 Introduction

In this paper we introduce and illustrate a graph calculus for deciding the positive identities and inclusions of fork algebras, i.e., those having no occurrences of complementation.

Relation algebras [15] are appropriate to formalize some aspects of program methodology despite their limitations on expressive and proof powers. One of the motivations for extending the relation algebraic formalism is capturing important notions linked to *storing* and *retrieving* of data, which is beyond its range. An operator introduced to this end is the *fork* operator [8], which is induced by a given injective coding function ¤, so that $b¤c$ can be regarded as an encoding of the ordered pair (b, c). Given (binary) relations X and Y on a base set U, by applying fork to X and Y we obtain the relation $\{(a, b¤c) : (a, b) \in X \text{ and } (a, c) \in Y\}$.

For instance, given relations X_1, X_2, X_3, X_4 on a set U, fork can be used to *store* the coded result of the application of X_1, X_2, X_3, X_4 to a single element $a \in U$. That is, a pair (a, b) belongs to the relation obtained by iterated application of fork to relations X_1, X_2, X_3, X_4 on U, iff there are b_1, b_2, b_3, $b_4 \in U$ such that $b = b_1¤(b_2¤(b_3¤b_4))$ and $(a, b_1) \in X_1, (a, b_2) \in X_2, (a, b_3) \in X_3, (a, b_4) \in$

* Research partially sponsored by CNPq (grants 301163/91-0, 471608/03-3 and 301526/2005-2), FAPERJ (grants E-26/131.180/2003, E-26/152.395/2002, APQ-1 E-26/ 170.335/2006, PRONEX E-26/171.536/2006 and Prociência) and FAPESP (grant ConsRel 2004/14107-2).

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 152–163, 2009.

X_4. Fork can also be used to define projection operators to *retrieve* the data stored. For instance, one can define a new constant operator π_3^4 that outputs the third coordinate of a quadruple, i.e., a pair (a, b) belongs to the relation π_3^4 iff there are $a_1, a_2, a_3, a_4 \in U$ such that a is the encoding $a_1 \natural (a_2 \natural (a_3 \natural a_4))$ and b is a_3.

Abstractly, relation algebras can be defined by a set of identities specifying the behavior of the Boolean and Peircean operators as follows. The former operators behave as in Boolean algebras. The latter operators behave as in involuted monoid theory. One also adds an identity expressing a geometric aspect of the interaction of Boolean and Peircean operators [9,14]. Fork algebras may be defined by extending relation algebras with a new binary operator $\triangledown$ (fork) and the following three identities:

(a) $(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \triangledown (\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \sqsubseteq \mathsf{I}$,
(b) $(r \triangledown s) \circ (t \triangledown q)^{\mathsf{T}} = (r \circ t^{\mathsf{T}}) \sqcap (s \circ q^{\mathsf{T}})$,
(c) $(r \circ (\mathsf{I} \triangledown \mathsf{E})) \sqcap (s \circ (\mathsf{E} \triangledown \mathsf{I})) = r \triangledown s$.

One of the most important characteristics of the fork algebraic apparatus is that it provides algebraic proofs of interesting program properties, such as input-output specification of programs, including refinement and abstraction; program behavior, including non-determinism and parallelism; program design strategies, including case analysis and divide-and-conquer, etc. [8]. Indeed, fork algebras are provided with an algebraic language where properties of program (schemata) can be expressed as equations and inferred from other equationally specified properties merely by replacing equals by equals.

Although the validity of some identities is easily established, this is not true in general: the equational theories of relation and fork algebras are rather complex [16]. Indeed, some non-algebraic mechanisms have been developed to cope with the problem of establishing identities involving relations [4,13]. This paper is a contribution in this line of development.

As a first step to overcome the difficulties mentioned above, we will introduce a formal calculus whose formulas are graphs (defining relations) and whose rules are used to derive graphs from graphs. We will prove that this calculus is sound, complete and decidable for graph inclusions. The graphical system can be directly applied to the positive fork algebraic inclusions and identities. Positive fork terms correspond to graphs and the graphical apparatus can be used to decide the equalities and identities in a playful manner.

In Section 2 we describe the positive fork language $+\mathsf{FL}$, its formal syntax and semantics, as well as some examples of equations that are valid. In Section 3 we introduce the positive fork calculus with graphs $+\mathsf{FG}$. It is built as an extension of the positive fork language hinging on an adequate notion of a graph labeled with fork terms. We also introduce inference rules to transform graphs to graphs illustrating their application. In Section 4 we establish the central metamathematical results of soundness and completeness as well as decidability. These results can be transferred directly to the the positive fork language. Section 5 closes the paper with some remarks and directions for future work.

2 Positive Fork Language

The positive fork relational language $+\mathsf{FL}$ is a variant of the positive relational language treated in [5,6] by introducing a new binary operator, called *fork* [8] and restricting semantics to structured models (cf. below).

The $+\mathsf{FL}$ *terms*, typically denoted R, S, T, are generated from the set of relational variables $\textsc{Rvar} = \{r_i : i \in \omega\}$ by applying the relational operators E, I, $^{\mathsf{T}}$, $\sqcap$, $\sqcup$, $\circ$, and $\triangledown$, according to the grammar $R ::= r_i \mid \mathsf{E} \mid \mathsf{I} \mid R^{\mathsf{T}} \mid R \sqcap S \mid R \sqcup S \mid R \circ S \mid R \triangledown S$. The *positive fork relational inclusions* and *equalities* are the expressions of the forms $R \sqsubseteq S$ and $R = S$, respectively.

A *structured universe* is a pair $(M, \boxtimes)$, where $M \neq \emptyset$ and $\boxtimes : M \times M \to M$ is an injective operation. Intuitively, $a \boxtimes b \in M$ codifies the pair (a, b) of elements of M. The *fork* induced by $\boxtimes$ in a structured universe $(M, \boxtimes)$ is the binary operation $\triangledown_{\boxtimes}$ on relations on M given by $R \triangledown_{\boxtimes} S := \{(a, b \boxtimes c) \in M \times M : aRb \text{ and } aSc\}$.

A *structured model* is a triple $\mathfrak{M} = (M, \boxtimes, r_i^{\mathfrak{M}})_{i \in \omega}$, where $(M, \boxtimes)$ is a structured universe and $r_i^{\mathfrak{M}} \subseteq M \times M$ for every $i \in \omega$. The *meaning* $[\![R]\!]_{\mathfrak{M}}$ of a term R in a structured model $\mathfrak{M}$ is defined similarly to the relational case (excluding all references to the empty relation and to complementation) together with an additional clause to treat the fork operator. Formally, given a structured model $\mathfrak{M} = (M, \boxtimes, r_i^{\mathfrak{M}})_{i \in \omega}$, symbols E and I are interpreted, respectively, as the relations $M \times M$ and $\{(a, a) : a \in M\}$; symbols $\sqcap$, $\sqcup$, $^{\mathsf{T}}$ and $\circ$ as intersection, union, conversion and composition of relations, respectively; and the meaning of a fork term $R \triangledown S$ is defined by $[\![R \triangledown S]\!]_{\mathfrak{M}} ::= [\![R]\!]_{\mathfrak{M}} \triangledown_{\boxtimes} [\![S]\!]_{\mathfrak{M}}$.

Validity of inclusions and equalities are defined as usual. As examples of valid formulas we have the three formulas (a), (b) and (c) taken as axioms for fork algebras in Section 1. In the sequel, we will introduce the positive fork graph calculus to prove valid inclusions.

3 Positive Fork Graph Calculus

In the positive graph relational calculus $+\mathsf{RG}$ [5,6], relations are represented by (directed pseudo multi) graphs having two distinguished nodes and arcs labeled by positive relational terms. In the positive fork graph calculus $+\mathsf{FG}$, we shall label arcs with positive fork relational terms and the graphs will represent binary relations on structured universes. We would like to distinguish ordinary nodes from those in the range of a star function.

We consider a fixed set $\textsc{Inod} = \{x_n : n \in \omega\}$ of *nodes*, typically denoted by x, y, z, u, v, w. Given set $N \subseteq \textsc{Inod}$, a *node equation* on N is a triple (u, v, w), denoted $u \star v \rightarrowtail w$, with $u, v, w \in N$; here w is the *star* of u and v, which are, respectively, *the first* and *the second components* of w. A *table* on N is a set T of node equations on N. With respect to a table T, we call a node w *structured* iff there is some node equation $u \star v \rightarrowtail w$ in T, otherwise, we call it *atomic*.

Graphically, we represent nodes by dots and node equations by two-source arrows pointing to the structured node and displaying its first and second components: a single line indicates the first component whereas a double line

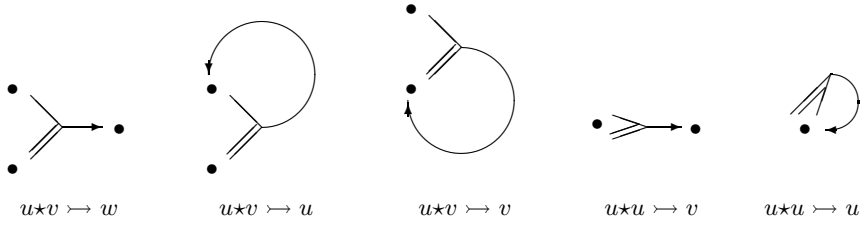


Fig. 1. Graphical representation of node equations

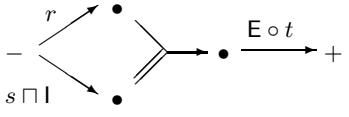


Fig. 2. Slice S with a structured node

indicates the second one. For instance, node equation $u\star v \rightarrowtail w$, where u, v, w are pairwise distinct, is represented in Figure 1.

An *arc* is a triple (u, R, v), denoted uRv, where u, v are nodes, and R is a $+\mathsf{FL}$ term. Graphically, we represent arcs by labeled arrows linking nodes.

A *slice* is a structure $S = (N, T, A, x, y)$, where N is a non-empty set of nodes, T is a table on N, $A \subseteq N \times \mathsf{Trm} \times N$ is a set of labeled *arcs* (Trm is the set of $+\mathsf{FL}$ terms), and x, y are (not necessarily distinct) nodes in N, called *input* and *output*, respectively. Graphically, we represent slices by directed arc-labeled pseudo multi graphs with distinguished nodes x, y represented by $-, +$, respectively. For instance, the slice $S = (\{x, u, v, w, y\}, \{u\star v \rightarrowtail w\}, \{xru, x(s \sqcap \mathsf{I})v, w(\mathsf{E} \circ t)y\}, x, y)$, having a structured node, is represented in Figure 2.

A *positive fork graph*, or simply a *graph*, typically denoted by G, H, is a finite non-empty set of slices. A graph can be represented by the juxtaposition of the representation of its slices. The $+\mathsf{FG}$ *inclusions* and *equalities* are expressions of the forms $G \sqsubseteq H$ and $G = H$, respectively.

The semantics of slices and graphs are based on binary relations. First, given a slice $S = (N, T, A, x, y)$ and a structured model $\mathfrak{M}$ with universe M, an $\mathfrak{M}$-*assignment* for S is a function $g : N \to M$ such that $(gu, gv) \in [\![R]\!]_{\mathfrak{M}}$, for every arc uRv in A, and $gu \leftthreetimes gv = gw$, for every node equation $u\star v \rightarrowtail w$ in T. Now, the *meaning* of a slice S in a model $\mathfrak{M}$ is the subset $[\![S]\!]_{\mathfrak{M}}$ of $M \times M$ defined by $(a, b) \in [\![S]\!]_{\mathfrak{M}}$ iff $gx = a, gy = b$, for some $\mathfrak{M}$-assignment g for S. The *meaning of graph* $G = (S_i)_{i\in I}$ in $\mathfrak{M}$ is $[\![G]\!]_{\mathfrak{M}} = \bigcup_{i\in I} [\![S_i]\!]_{\mathfrak{M}}$.

A $+\mathsf{FG}$ inclusion $G \sqsubseteq H$ *holds* in a model $\mathfrak{M}$, denoted $\mathfrak{M} \models G \sqsubseteq H$, iff $[\![G]\!]_{\mathfrak{M}} \subseteq [\![H]\!]_{\mathfrak{M}}$. It is *valid*, denoted $\models G \sqsubseteq H$, iff it holds in every model. Analogously, we can define truth and validity for $+\mathsf{FG}$ equalities and prove results similar to those described in Section 2 for $+\mathsf{FL}$ inclusions and equalities. In particular, graphs G and H are called *equivalent* iff $\models G = H$.

Table 1. Elimination/Introduction rules for transforming graphs

$$\mathsf{Unv}\ \frac{G \cup \{(N, A \cup \{u\mathsf{E}v\}, x, y)\}}{G \cup \{(N, A, x, y)\}} \quad \mathsf{Idn}\ \frac{G \cup \{(N, A \cup \{u\mathsf{I}v\}, x, y)\}}{G \cup \{(N\frac{u}{v}, A\frac{u}{v}, x\frac{u}{v}, y\frac{u}{v})\}}$$

$$\mathsf{Cnv}\ \frac{G \cup \{(N, A \cup \{uR^{\mathsf{T}}v\}, x, y)\}}{G \cup \{(N, A \cup \{vRu\}, x, y)\}} \quad \mathsf{Int}\ \frac{G \cup \{(N, A \cup \{uR \sqcap Sv\}, x, y)\}}{G \cup \{(N, A \cup \{uRv, uSv\}, x, y)\}}$$

$$\mathsf{Uni}\ \frac{G \cup \{(N, A \cup \{uR \sqcup Sv\}, x, y)\}}{G \cup \{(N, A \cup \{uRv\}, x, y), (N, A \cup \{uSv\}, x, y)\}}$$

$$\mathsf{Cmp}\ \frac{G \cup \{(N, A \cup \{uR \circ Sv\}, x, y)\}}{G \cup \{(N \cup \{w\}, A \cup \{uRw, wSv\}, x, y)\}} \quad \text{if } w \notin N$$

(a) Relational rules

$$\mathsf{Frk}\ \frac{G \cup \{(N, T, A \cup \{uR \triangledown Sv\}, x, y)\}}{G \cup \{(N \cup \{v_1, v_2\}, T \cup \{v_1 \star v_2 \rightarrowtail v\}, A \cup \{uRv_1, uSv_2\}, x, y)\}} \quad \text{if } v_1, v_2 \notin N$$

(b) Fork rule

The deductive apparatus of +FG is given by a set of graph transforming rules. Some rules transform a graph into an equivalent one (usually used to put a graph in normal form), while another rule will be used to compare graphs (usually in normal form). We will use the *node substitution* notation $\frac{u}{v}$ for replacing u by v, which we extend naturally to pairs and triples as well as sets; e.g., for a set A of arcs, we put $A\frac{u}{v} := \{w\frac{u}{v}Rz\frac{u}{v} : wRz \in A\}$.

The *transformation rules* are given in Tables 1, 2 and 3. Rules in Table 1(a) cover the relational part of the fork graphs and the rule in Table 1(b) covers similarly the fork operator. The star rules in Table 2 concern the graph tables. The rules in these three tables can be applied in both directions. In fact, each one of these rules is an abbreviation for two rules: downward and upward. The rules in Table 1 allow the *elimination* (downwards) and the *introduction* (upwards) of the operators. Table 3 presents the capital rule for comparing graphs.

We will explain each bidirectional rule in the downward direction. Each rule in Tables 1 and 2 states that the meaning of graph does not change when applying the local transformation specified in the rule, leaving the rest of graph untouched. Soundness will follow from the explanations.

Rules in Table 1(a) concern the relational part of the fork graphs, being similar to those of +RG [6]. Rule Unv allows erasing an arc labeled by E from a slice. Rule Idn allows one to erase an arc $u\mathsf{I}v$ and a node u, renaming nodes and redirecting arcs accordingly. Rule Cnv allows replacing an arc $uR^{\mathsf{T}}v$ by vRu. Rule Int allows one to replace an arc $uR \sqcap Sv$ by two others uRv and uSv. Rule Uni allows one to replace a slice C having an arc $uR \sqcup Sv$, by two other slices C_R and C_S,

Table 2. Star rules for transforming graphs

$$\mathsf{Tot}\ \frac{G \cup \{(N,T,A,x,y)\}}{G \cup \{(N \cup \{w\}, T \cup \{u\star v \rightarrowtail w\}, A, x, y)\}} \quad \text{if } w \notin N$$

$$\mathsf{Fnc}\ \frac{G \cup \{(N,T \cup \{u\star v \rightarrowtail w_1, u\star v \rightarrowtail w_2\}, A, x, y)\}}{G \cup \{(N\frac{w_1}{w_2}, (T \cup \{u\star v \rightarrowtail w_1\})\frac{w_1}{w_2}, A\frac{w_1}{w_2}, x\frac{w_1}{w_2}, y\frac{w_1}{w_2})\}}$$

$$\mathsf{Inj}\ \frac{G \cup \{(N,T \cup \{u_1\star v_1 \rightarrowtail w, u_2\star v_2 \rightarrowtail w\}, A, x, y)\}}{G \cup \{((N\frac{u_1}{u_2})\frac{v_1}{v_2}, [(T \cup u_1\star v_1 \rightarrowtail w)\frac{u_1}{u_2}]\frac{v_1}{v_2}, (A\frac{u_1}{u_2})\frac{v_1}{v_2}, (x\frac{u_1}{u_2})\frac{v_1}{v_2}, (y\frac{u_1}{u_2})\frac{v_1}{v_2})\}}$$

Table 3. Homomorphism rule GrCvr for transforming graphs

$$\mathsf{GrCvr}\ \frac{G}{H} \quad \text{if } G \leftarrow H$$

obtained from C by replacing the arc $uR \sqcup Sv$ by a new arc: uRv for C_R and uSv for C_S. Rule Cmp allows one to replace an arc $uR \circ Sv$ by two others, uRw and wSv, with a new node w.

Table 1(b) presents the Frk rule concerning the fork operator: it allows one to replace an arc $uR \triangledown Sv$ by two other arcs uRv_1 and uSv_2, with two new nodes v_1 and v_2.

Table 2 presents the rules concerning graph tables. Rule Tot allows one to add a new node as the star of given nodes. Rule Fnc allows one to identify nodes that are the star of the same pair of nodes. Rule Inj allows one to identify nodes that are the first and the second components, respectively, of a node. Each one of these rules is sound since ¤ is a total, injective function.

Table 3 presents the capital rule GrCvr: it allows one to infer a graph from one covered under homomorphism. The notions involved are natural extensions of the +RG case [6].

Given slices $S = (N,T,A,x,y)$ and $S' = (N',T',A',x',y')$, a *homomorphism* from S' to S (denoted $\theta : S' \rightarrow S$) is a function $\theta : N' \rightarrow N$ that preserves the slice structure: $\theta w\star\theta u \rightarrowtail \theta v \in T$, for every node equation $w\star u \rightarrowtail v$ in T'; $\theta uR\theta v \in A$, for every arc uRv in A'; $\theta x' = x$ and $\theta y' = y$. Given graphs G and H, we say that H *covers* G (denoted $G \leftarrow H$) iff, for each slice S of G, there is a slice S' of H and a homomorphism $\theta : S' \rightarrow S$.

Rule GrCvr can be applied only downwards, but a special case of GrCvr can be applied in both directions, namely, the derived rule ErUn for erasing useless nodes presented in Table 4. A node is *useless* in a slice iff it is not distinguished and does not occur in its arcs nor in its table.

An inclusion $G \sqsubseteq H$ is a +FG *theorem*, denoted $\vdash G \sqsubseteq H$, iff H can be obtained from G by applications of the inference rules. We will call graphs G

Table 4. Derived rule ErUn for erasing useless nodes

$$\mathsf{ErUn}\ \frac{N \cup w, T, A, x, y}{N, T, A, x, y} \quad \text{if } w \text{ is useless}$$

and H *provably equivalent* iff $\vdash G \sqsubseteq H$ and $\vdash H \sqsubseteq G$. We call a proof *normal* iff it consists of applications of elimination, star and ErUn rules, followed by a single application of the GrCvr rule, followed by applications of introduction, star and ErUn rules.

To illustrate the system in action, we associate to a fork relational term R its graph $G_R := \{(\{x, y\}, \emptyset, \{xRy\}, x, y)\}$. Then, it is clear that $[\![G_R]\!]_{\mathfrak{M}} = [\![R]\!]_{\mathfrak{M}}$, for every model $\mathfrak{M}$. So, we can reduce inclusions between fork relational terms to inclusions between their associated graphs: $\models R \sqsubseteq S$ iff $\vdash G_R \sqsubseteq G_S$. A graph proof of the $+\mathsf{FG}$ axiom (a) is indicated in Figure 3.

$+\mathsf{FG}$ also can model a kind of parallel product through the introduction of the operator *cross*. Given a pair of relations X and Y on a set U, by applying cross to X and Y we obtain the relation $\{(\boxtimes ab, \boxtimes cd) : (a, c) \in X \text{ and } (b, d) \in Y\}$. A fork algebraic definition of cross is given by [8]: $r \otimes s := ((\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ r) \triangledown ((\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ s)$. In [8], $\otimes$ is extensively used to prove identities as

(d) $(r \triangledown s) \circ (t \otimes u) = (r \circ t) \triangledown (s \circ u)$,

describing the iterated behaviour of the algebraic operators. Figure 4 contains an equational proof of (d).

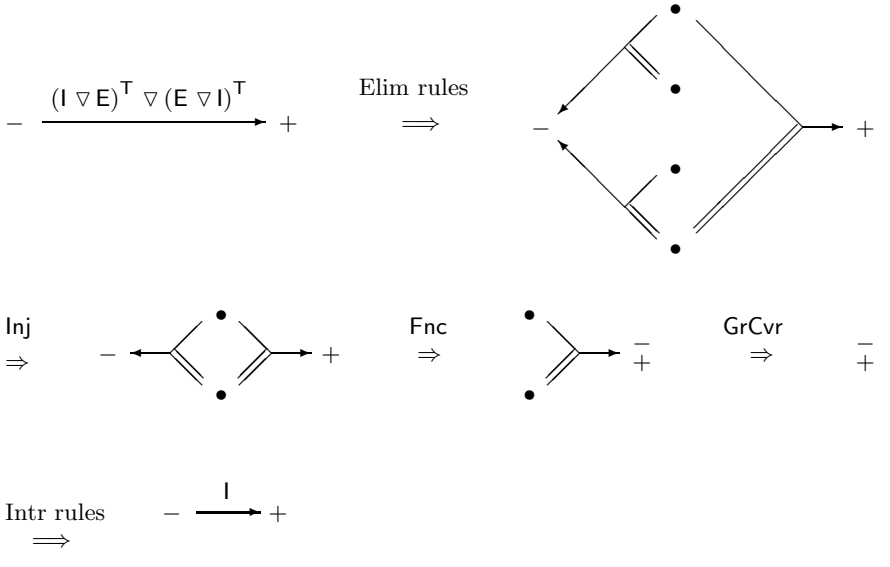


Fig. 3. Graph proof of fork axiom (a)

$$
\begin{aligned}
(r \triangledown s) \circ (t \otimes u) &= (r \triangledown s) \circ (t \otimes u)^{\mathsf{T}^{\mathsf{T}}} \\
&= (r \triangledown s) \circ [((\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t) \triangledown ((\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u)]^{\mathsf{T}^{\mathsf{T}}} \\
&= (r \triangledown s) \circ [[(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t \circ (\mathsf{I} \triangledown \mathsf{E})] \sqcap [((\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u \circ (\mathsf{E} \triangledown \mathsf{I}))]]^{\mathsf{T}^{\mathsf{T}}} \\
&= (r \triangledown s) \circ [[(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t \circ (\mathsf{I} \triangledown \mathsf{E})]^{\mathsf{T}} \sqcap [(\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u \circ (\mathsf{E} \triangledown \mathsf{I})]^{\mathsf{T}}]^{\mathsf{T}} \\
&= (r \triangledown s) \circ [[(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t^{\mathsf{T}} \circ (\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}^{\mathsf{T}}}] \sqcap [(\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u^{\mathsf{T}} \circ (\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}^{\mathsf{T}}}]^{\mathsf{T}} \\
&= (r \triangledown s) \circ [[(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t^{\mathsf{T}} \circ (\mathsf{I} \triangledown \mathsf{E})] \sqcap [(\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u^{\mathsf{T}} \circ (\mathsf{E} \triangledown \mathsf{I})]]^{\mathsf{T}} \\
&= (r \triangledown s) \circ [[(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t^{\mathsf{T}}] \sqcap [(\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u^{\mathsf{T}}]]^{\mathsf{T}} \\
&= [r \circ [(\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}} \circ t^{\mathsf{T}}]^{\mathsf{T}}] \sqcap [s \circ [(\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}} \circ u^{\mathsf{T}}]^{\mathsf{T}}] \\
&= [r \circ [t^{\mathsf{T}^{\mathsf{T}}} \circ (\mathsf{I} \triangledown \mathsf{E})^{\mathsf{T}^{\mathsf{T}}}]] \sqcap [s \circ [u^{\mathsf{T}^{\mathsf{T}}} \circ (\mathsf{E} \triangledown \mathsf{I})^{\mathsf{T}^{\mathsf{T}}}]] \\
&= [r \circ [t \circ (\mathsf{I} \triangledown \mathsf{E})]] \sqcap [s \circ [u \circ (\mathsf{E} \triangledown \mathsf{I})]] \\
&= (r \circ t) \triangledown (s \circ u).
\end{aligned}
$$

Fig. 4. Equational proof of (d)

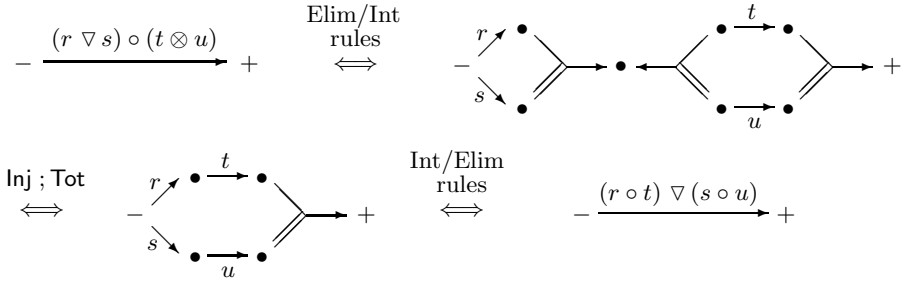


Fig. 5. Graph proof of (d)

Figure 5 presents an alternative graph proof, based on the derived rule Paral in Table 5, whose graph derivation is in Figure 6.

4 Metamathematics of the Fork Graph Calculus

In this section, we prove soundness, completeness and decidability of the positive fork graph calculus with respect to the +FG valid inclusions. The approach presented here is a substantial extension of the one given in [5], for +RG.

Soundness of +FG is an immediate consequence of the Lemma 1.

Lemma 1. *Consider graphs G and H. If H is obtained from G by applications of the rules in Tables 1, 2 and 4, then $\models G = H$. If H is obtained from G by the rule* GrCvr *then $\models G \sqsubseteq H$.*

For the remaining results, we will define a normal form for graphs and show that inclusion of graphs can be reduced to inclusion of their normal forms.

Table 5. Derived introduction/elimination rule for $\otimes$

Paral

$$\frac{N, T, A \cup \{uR \otimes Sv\}, x, y}{N \cup \{w_1, w_2, w_3, w_4\}, T \cup \{w_1 \star w_3 \rightarrowtail u, w_2 \star w_4 \rightarrowtail v\}, A \cup \{w_1 R w_2, w_3 S w_4\}, x, y}$$

if $w_1, w_2, w_3, w_4 \notin N$

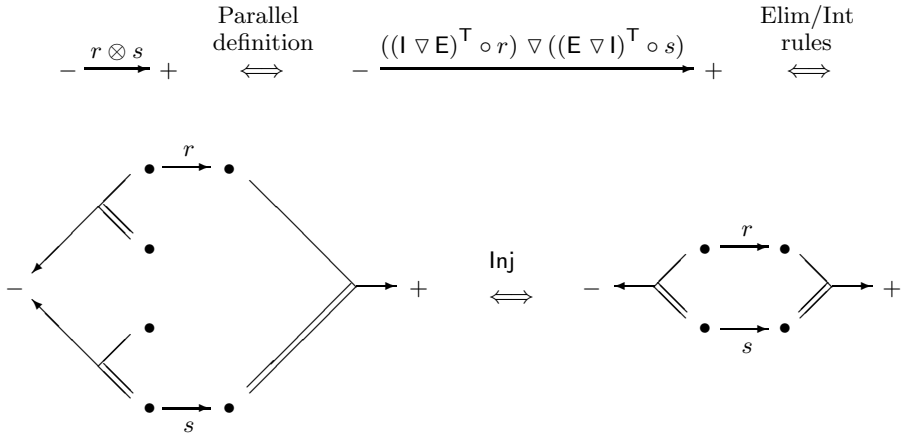


Fig. 6. Derivation of graph rule Paral

A table T induces a relation $\star_T$ on $(N \times N) \times N$ such that $(u, v) \star_T w$ iff $u \star v \rightarrowtail w \in T$. We call a table *functional* or *injective* iff the induced relation is functional or injective, respectively.

A *star-path* in a slice is a sequence $(u_1, v_1, \ldots, u_n, v_n, w)$ of nodes such that $u_n \star v_n \rightarrowtail w \in T$ and for each $i, 1 \leq \ldots i \ldots \leq n-1$, $u_i \star v_i \rightarrowtail u_{i+1} \in T$ or $u_i \star v_i \rightarrowtail v_{i+1} \in T$. A *star-cycle* in a slice is a sequence $(u_1, v_1, \ldots, u_n, v_n)$ of nodes such that $u_n \star v_n \rightarrowtail u_1 \in T$ or $u_n \star v_n \rightarrowtail v_1 \in T$, and for each $i, 1 \leq \ldots i \ldots \leq n-1$, $u_i \star v_i \rightarrowtail u_{i+1} \in T$ or $u_i \star v_i \rightarrowtail v_{i+1} \in T$.

Now, we introduce the notion of essential node, which will play a central role in the definition of normal form for graphs. A node v is *essential* in a slice S if it is n-essential in S, for some $n \in \mathbb{N}$. A node v is 0-*essential* in a slice S if v is distinguished in S, or v is an extreme of an arc in S, or v is an element of a star-cycle in S. A node v is n-*essential* in a slice S, for $n > 0$, if there is a star-path $(u_1, v_1, \ldots, u_n, v_n, w)$ in S such that v is not m-essential in S, for $m < n$, w is 0-essential in S, and $v = u_1$ or $v = v_1$.

We call a graph G *basic* iff every arc in G is labeled by a relational variable, *functionally injective* iff every table in G is functional and injective, and *lean* iff every node in G is essential. We say that G is in *normal form* iff G is basic, functionally injective and lean.

Lemma 2 (Conversion to normal form). *Every graph G can be effectively converted (by elimination, star and* ErUn *rules) into a provably equivalent graph νG in normal form.*

Proof. By applying elimination rules we reduce G to a basic G_1 (by induction on the number of operators in the labels of arcs in the graph). By applying rules Fnc and Inj we reduce G_1 to a functionally injective G_2 (by induction on the number of nodes in the slices of the graph). We can reduce G_2 to a lean νG, noting that, as the table of G_2 is injective, a non-essential node occurs in a star-path or is useless. If it occurs in star-path, this star-path has a sink which is not a 0-essential node and occurs in exactly one node-equation, whence this sink can be eliminated by rule Tot. In the latter case, we use rule ErUn. We thus effectively convert G into νG in normal form, they are provably equivalent since the rules used are reversible.

Thus, by soundness, we can reduce inclusions to inclusion of normal forms.

Corollary 1 (Reduction). *Given fork graphs G and H, $\models G \sqsubseteq H$ iff $\models \nu G \sqsubseteq \nu H$.*

The next lemma is our main tool for establishing the remaining results. Its proof is based on a construction of a model induced by a slice in normal form.

Lemma 3. *Given fork graphs G and H in normal form, if $\models G \sqsubseteq H$, then H covers G.*

Proof. Assume $\models G \sqsubseteq H$. Let $S = (N, T, A, x, y)$ be a slice of G. Construct structured model $\mathfrak{M}_S = (N^{\maltese}, \maltese, r_i^{\mathfrak{M}_S})_{i\in\omega}$ as follows:

- $N^{\maltese} := \bigcup_{k\in\mathbb{N}} N_k$, with $N_0 := N$ and $N_{k+1} := N_k \cup \{(u,v) \in N_k \times N_k : \forall w \in N_k (u{\star}v \rightarrowtail w \notin T)\}$,
- $r_i^{\mathfrak{M}_S} := \{(u,v) \in N \times N : ur_iv \in A\}$,
- $u \maltese v := \begin{cases} w & \text{if } u{\star}v \rightarrowtail w \in T \\ (u,v) & \text{otherwise.} \end{cases}$

It is easy to see that $(x,y) \in [\![G]\!]_{\mathfrak{M}_S}$.

Also, $\mathfrak{M}_S$ is a model, since by construction, $\maltese$ is total as well as injective and is functional. Since $\models G \sqsubseteq H$, we have $(x,y) \in [\![H]\!]_{\mathfrak{M}_S}$. Thus, consider a slice $S' = (N', A', T', x', y')$ of H and an $\mathfrak{M}_S$-assignment $g : N' \to N^{\maltese}$ with $gx' = x$, $gy' = y$.

We claim that the range of g is a subset of N. In fact, given $v \in N'$, we have that v is essential, since H is in normal form. If v is distinguished in S', or an extreme of an arc in S', then $gv \in N$. If v is in a star-cycle in S', then $gv \in N$, since the nodes introduced in the construction of $\mathfrak{M}_S$ do not belong to cycles. If v is in a star-path $(u_1, v_1, \ldots, u_n, v_n, w)$ in S' such that $v = u_1$ or $v = v_1$ and w is a 0-essential node in S', then, by previous cases, $gw \in N$. Since $u_n{\star}v_n \rightarrowtail w \in T'$, we have $gu_n \maltese gv_n = gw$. Hence, $gu_n{\star}gv_n \rightarrowtail gw \in T$. So, $gu_n, gv_n \in N$. Applying the same reasoning backwards, we obtain $gv \in N$.

To show that g is a homomorphism from S' to S, it remains to see that g preserves arcs and tables, but this is clear, because g is an $\mathfrak{M}_S$-assignment.[1]

We thus immediately have our central equivalences.

Proposition 1. *Given fork graphs G and H, the following assertions are equivalent.*

1. *The inclusion $G \sqsubseteq H$ is valid: $\models G \sqsubseteq H$.*
2. *The inclusion $\nu G \sqsubseteq \nu H$ is valid: $\models \nu G \sqsubseteq \nu H$.*
3. *νH covers νG: $\nu G \leftarrow \nu H$.*
4. *The inclusion $G \sqsubseteq H$ is a theorem: $\vdash G \sqsubseteq H$.*

We thus have completeness (of normal) proofs and decidability.

Theorem 1. *Given a fork graph G and H, consider the inclusion $G \sqsubseteq H$.*
(a) If $G \sqsubseteq H$ is valid, then there is a normal proof of H from G.
(b) The inclusion $G \sqsubseteq H$ is valid iff νH covers νG.

5 Perspectives

We have presented a graph calculus for proving and deciding the positive identities and inclusions of fork algebras. In this calculus, formulas are graphs and the rules derive graphs from graphs. This calculus is sound, complete and decidable for graph inclusions. We have illustrated how this graphical apparatus can be directly applied to the positive fork algebraic inclusions and identities. Our fork calculus considered structured universes with a total injective function ¤. A natural extension would be providing sound and complete calculi for structured universes with weaker restrictions imposed on ¤.

Proofs of inclusions and identities from hypotheses are also interesting. Extending our system to cope with this non-decidable case will involve more elaborated work.

Pictures have been proposed as a tool to help investigating and applying relational formalisms. Here, we mention three main lines of research. The approach based on the *theory of allegories* [1,2,3,4,10], the approach based on the *rewriting systems* [11,12,13], and the *logic systematic* approach [3,5,6]. Each one of these approaches has its own flavor, techniques of investigations and line of results. Nevertheless, they are not completely disjoint sharing many characteristics whose interactions deserve further investigation. The work reported here may also be viewed as a first contribution in this direction in that we extend the logic systematic approach to the positive fork language. We thus provide a basis for a new formalism, which is not only more widely applicable but also provides a common denominator of the above three lines of investigation.

[1] For a node equation $u \star v \rightarrowtail w \in T$', we have $gu ¤ gv = gw$, and as $gw \in N$, we have $gu \star gv \rightarrowtail gw \in T$. For an arc $ur_iv \in A'$, we have $(gu, gv) \in [\![r_i]\!]_{\mathfrak{M}_S}$, and by the definition of $\mathfrak{M}_S$, one has $gur_igv \in A$.

References

1. Brown, C., Hutton, G.: Categories, allegories and circuit design. In: 9th IEEE Symp. Logic in Computer Science, pp. 372–381. IEEE Computer Society Press, Los Alamitos (1994)
2. Brown, C., Jeffrey, A.: Allegories of circuits. In: Proc. Logical Foundations of Computer Science, St. Petersburg, pp. 56–68 (1994)
3. Curtis, S., Lowe, G.: A graphical calculus. In: Möller, B. (ed.) MPC 1995. LNCS, vol. 947, pp. 214–231. Springer, Heidelberg (1995)
4. Curtis, S., Lowe, G.: Proofs with graphs. Sci. Comput. Program. 26, 197–216 (1996)
5. de Freitas, R.P., Veloso, P.A.S., Veloso, S.R.M., Viana, P.: Reasoning with graphs. ENTCS, vol. 165, pp. 201–212 (2006)
6. de Freitas, R.P., Veloso, P.A.S., Veloso, S.R.M., Viana, P.: On positive relational calculi. Logic J. IGPL 15, 577–601 (2006)
7. Freyd, P.J., Scedrov, A.: Categories, Allegories. North-Holland, Amsterdam (1990)
8. Frias, M.: Fork Algebras in Algebra, Logic and Computer Science. World Scientific, Singapore (2002)
9. Hirsch, R., Hodkinson, I.: Relation Algebras by Games. Elsevier, Amsterdam (2002)
10. Hutton, G.: A relational derivation of a functional program. In: Lecture Notes of the STOP Summer School on Constructive Algorithms, Ameland (1992)
11. Kahl, W.: Algebraic graph derivations for graphical calculi. In: D'Amore, F., Franciosa, P.G., Marchetti-Spaccamela, A. (eds.) WG 1996. LNCS, vol. 1197, pp. 224–238. Springer, Heidelberg (1997)
12. Kahl, W.: Relational treatment of term graphs with bound variables. Logic J. IGPL 6, 259–303 (1998)
13. Kahl, W.: Relational matching for graphical calculi of relations. Inform. Sciences 119, 253–273 (1999)
14. Maddux, R.D.: Relation Algebras. Elsevier, Amsterdam (2006)
15. Maddux, R.D.: Relation-algebraic semantics. Theor. Comput. Sci. 160, 1–85 (1996)
16. Mikulász, S., Sain, I., Simon, A.: Complexity of the equational theory of relation algebras with projection elements. Bull. Sect. Logic Univ. Lódź 21, 103–111 (1992)

Games on Strings with a Limited Order Relation*

Elisabetta De Maria, Angelo Montanari, and Nicola Vitacolonna

Department of Mathematics and Computer Science, University of Udine, Italy

Abstract. In this paper, we show how Ehrenfeucht-Fraïssé games can be successfully exploited to compare (finite) strings. More precisely, we give necessary and sufficient conditions for Spoiler/Duplicator to win games played on finite structures with a limited order relation, that lies in between the successor relation and the usual (linear) order relation, and a finite number of unary predicates. On the basis of such conditions, we outline a polynomial (in the size of the input strings) algorithm to compute the "remoteness" of a game and to determine the optimal strategies/moves for both players.

1 Introduction

Comparison games are mainly used to prove inexpressibility results or to establish normal forms for logics [1,2,3]. We take a different point of view: given two (finite) structures, we use comparison games to determine how and where they differ. Such an approach has a variety of applications in data matching. For instance, it can be exploited in biological sequence comparison, where left-to-right matches turned out to be inadequate. In [4], Montanari et al. have considered structures provided with the successor relation s: they define a criterion to measure the degree of similarity of labeled successor structures and develop an algorithm to compute it. As an alternative, one may think of replacing s with the linear order relation $<$; however, $<$ does not preserve locality and thus phenomena such as inversions (some parts may appear in a different order in homologous sequences) cannot be dealt with. In this paper, we assume a more general point of view by considering a relation that lies in between s and $<$. More precisely, we study first-order Ehrenfeucht-Fraïssé games (*EF-games* for short) played on structures with a limited ordering $<_p$, called *labeled* $<_p$*-structures*, which is defined as follows: given a pair of positions i and j, we have that $i <_p j$ if and only if $i < j$ and $j - i \leq p$. Relations s and $<$ can be recovered as special cases of $<_p$ for p equal to 1 and p greater than or equal to the maximum of the lengths of the two sequences, respectively. From a technical point of view, suitable abstractions are needed for describing winning strategies for EF-games on labeled $<_p$-structures, e.g., the distinction between rigid and elastic intervals, that make the proofs much more involved.

The *playground* for a *comparison game* (on strings) is defined by a pair of strings w, w'. A *round* consists in a Spoiler's move followed by a Duplicator's

* This work is partially supported by the MIUR project FIRB03-RBNE03B8KK.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 164–179, 2009.

one (hereafter we will abbreviate Spoiler as **I** and Duplicator as **II**). At each round, **I** chooses one of the two strings and a position in it; **II** replays choosing a position in the other string. A *configuration* is a "snapshot" of the playground at a given stage of the game. We will write $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ to indicate the configuration where $\mathbf{i}^n$ (resp., $\mathbf{j}^n$) is the tuple of positions already chosen in w (resp. w'). A game with q rounds on $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$ is denoted by $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$; a game that goes on until either **I** repeats a move (**II** wins) or the current position is not a partial isomorphism (**I** wins) is denoted by $\mathcal{G}((w, \mathbf{i}^n), (w', \mathbf{j}^n))$. In the following, we will write $\mathbf{I}(\mathcal{G}_q)$ (resp., $\mathbf{II}(\mathcal{G}_q)$) to state that **I** (resp., **II**) has a winning strategy in q rounds. The *remoteness* $R(\mathcal{G})$ of $\mathcal{G}$ is the minimum q such that $\mathbf{I}(\mathcal{G}_q)$ (if such a q does not exist, we assume the remoteness to be infinite). A move by **I** in $\mathcal{G}$ is *optimal* if, whatever **II** replies, the new game $\mathcal{G}'$ has remoteness $R(\mathcal{G}') \leq R(\mathcal{G}) - 1$; similarly, **II**'s reply is optimal if $R(\mathcal{G}') \geq R(\mathcal{G}) - 1$ no matter how **I** has played. The algorithmic complexity of EF-games has not been thoroughly investigated. In [5], it is proved that, given two (finite) structures, determining whether $\mathbf{II}(\mathcal{G}_q(\mathfrak{A}, \mathfrak{B}))$ is PSPACE-complete when the vocabulary contains at least one binary and one ternary relation symbol. Efficient algorithms for EF-games on specific classes of structures (equivalence relations, trees, unary relation structures, Boolean algebras, and some natural extensions of them) are given in [6]. In this paper, we provide a characterization of EF-games on labeled $<_p$-structures that allows us to answer the following questions in polynomial time: given $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$, what is the remoteness of $\mathcal{G}((w, \mathbf{i}^n), (w', \mathbf{j}^n))$? What are the sets of **I**'s and **II**'s optimal moves?

In [4], Montanari et al. propose an algorithm to solve EF-games on labeled successor structures (LSSs) $\mathfrak{A}, \mathfrak{B}$ in $O(n \log n)$ time, where $n = |\mathfrak{A}| + |\mathfrak{B}|$. The solution consists in combining a local and a global strategy. More precisely, $\mathbf{II}(\mathcal{G}_q)$ iff the game is *q-locally safe* and *q-globally safe*. The former property ensures that corresponding distinguished positions have the same relative positions up to a threshold distance and that suitable corresponding neighborhoods spell equal substrings in both words, so that **II** can correctly play q rounds within such neighborhoods (local moves). The latter property is based on counting the multiplicity and the degree of scattering of substrings "falling far away" from distinguished positions. In the case of $<_p$, the global strategy remains essentially unchanged, while the local strategy is quite different. Let us suppose **I**'s moves are coerced to be local. We first provide a necessary and sufficient condition for **II** to win games in q rounds on structures devoid of unary predicates (*q-distance safety*). Roughly speaking, distinguished positions give rise to "rigid" and "elastic" intervals. Suppose that **I** and **II** play the game $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$. If **I** chooses a new position i_{n+1} inside one of the rigid intervals induced by a distinguished position $i_h \in \mathbf{i}^n$, then **II** must choose a new position j_{n+1} such that $j_{n+1} - j_h = i_{n+1} - i_h$. If the new position is chosen inside one of the elastic intervals induced by i_h, then **II** must reply by choosing a new position inside the elastic interval induced by j_h, but not necessarily at the same distance. This guarantees q-distance safety. To deal with unary predicates, we introduce an additional condition ($<_p$*-safety for q-colors*). The basic ingredient is a recursive

notion of *q-color*, which presents some similarities with the one given in [1]. $<_p$-safety for q-colors is guaranteed by constraining corresponding positions in $\mathbf{i}^n, \mathbf{j}^n$ to have the same q-color. As a matter of fact, for $p = 1$ the above conditions are equivalent to those for the successor relation s (as expected), while if $p \geq \max(|w|, |w'|)$, then $<_p$ coincides with $<$. Similarly to the case of LSSs, the global strategy essentially consists in comparing the *multiplicity* and the *scattering* of r-colors, with $0 \leq r \leq q-1$, in the portions of w and w' that are far away from $\mathbf{i}^n$ and $\mathbf{j}^n$, respectively.

The paper is organized as follows. Section 2 provides some preliminary definitions. Section 3 analyzes the basic case of local games on $<_p$-structures. Section 4 takes into consideration local games on labeled $<_p$-structures. Finally, global games on labeled $<_p$-structures are the subject of Sect. 5. Conclusions provide an assessment of the achieved results and outline future work directions. Detailed proofs and additional examples can be found in [7].

2 Basic Definitions

Let $\langle R, n\rangle$ denote a relation R with arity n and let $\tau = \{\langle R_1, n_1\rangle, \ldots, \langle R_k, n_k\rangle\}$ be a finite relational vocabulary. A *τ-structure* $\mathfrak{A}$ is a tuple consisting of a set A, called the *domain*, and an n_i-ary relation $R_i^A \subseteq A^{n_i}$ for each $\langle R_i, n_i\rangle \in \tau$. As we already did in the introduction, we denote τ-structures with symbols $\mathfrak{A}$, $\mathfrak{B}$, etc... In the following, we assume that every vocabulary implicitly contains a symbol $=$ interpreted as equality. Besides, we restrict our attention to finite structures expanded with constants. A structure $\mathfrak{A}$ with distinguished elements $a_1, \ldots, a_k \in A$ is denoted by $(\mathfrak{A}, \mathbf{a}^k)$. Given a structure $\mathfrak{A}$ and $A' \subseteq A$, the *substructure* of $\mathfrak{A}$ *induced by* A' is the structure with domain A' whose relations are the relations of $\mathfrak{A}$ restricted to A'.

Two structures $(\mathfrak{A}, \mathbf{a}^h)$ and $(\mathfrak{B}, \mathbf{b}^k)$, with $h, k \geq 0$, are *isomorphic*, $(\mathfrak{A}, \mathbf{a}^h) \cong (\mathfrak{B}, \mathbf{b}^k)$ for short, if $h = k$ and there is an isomorphism f between $\mathfrak{A}$ and $\mathfrak{B}$ such that $f(a_j) = b_j$, for $1 \leq j \leq k$. A *partial isomorphism* between $(\mathfrak{A}, \mathbf{a}^k)$ and $(\mathfrak{B}, \mathbf{b}^k)$, with $k \geq 0$, is an isomorphism of the substructures of $\mathfrak{A}$ and $\mathfrak{B}$ induced by $\mathbf{a}^k$ and $\mathbf{b}^k$, respectively. Since we are interested in structures associated with strings, we give further definitions for this special case. Let Σ be a fixed alphabet, $w \in \Sigma^*$, and $p \geq 1$. A *labeled $<_p$-structure* is a pair $(w, \mathbf{i}^n)$, where $w = (\{1, \ldots, |w|\}, <_p, (P_a)_{a\in\Sigma})$, with $i <_p j$ if and only if $0 < j - i \leq p$, for all $i, j \in \{1, \ldots, |w|\}$, and $i \in P_a$ if and only if $w[i] = a$ for all $i \in \{1, \ldots, |w|\}$, and $\mathbf{i}^n$ are distinguished positions $i_1, \ldots, i_n \in \{1, \ldots, |w|\}$. Moreover, given $i, j \in \{1, \ldots, |w|\}$, the *distance* $\delta(i, j)$ between $w[i]$ and $w[j]$ is $|i - j|$. The *k-distance* $\delta_k(i, j) : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ is a "truncated" distance defined as follows: $\delta_k(i, j) = \delta(i, j)$ if $\delta(i, j) \leq k$, $\delta_k(i, j) = \infty$ otherwise.

3 Local Games on $<_p$ Structures

In this section we consider the simplified case of local games on strings devoid of unary predicates. We provide necessary and sufficient conditions for $\mathbf{II}(\mathcal{G}_q)$,

for any given q. The first concept we introduce is the one of *pstep*, which can be used to measure the "signed distance" between two elements in terms of the number of intervals of length p separating them.

Definition 1. *Let $i, j, k, p \in \mathbb{N}$, with $i, j, p > 0$ and $k \geq p$. We define the function $pstep_k^{(p)}(i, j)$ as follows:*

$$pstep_k^{(p)}(i,j) = \begin{cases} 0 & \text{if } i = j \\ \lceil \frac{\delta(i,j)}{p} \rceil & \text{if } \delta_k(i,j) < \infty \text{ and } i < j \\ \lfloor -\frac{\delta(i,j)}{p} \rfloor & \text{if } \delta_k(i,j) < \infty \text{ and } i > j \\ \infty & \text{if } \delta_k(i,j) = \infty. \end{cases}$$

The next definition expresses *pstep-safety* of configurations (in the k-horizon). Condition (1) refers to internal positions, while conditions (2) and (3) deal with positions belonging to the prefix and the suffix, respectively.

Definition 2. *Let $p, k \in \mathbb{N}$, with $k \geq p$. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* pstep-safe *in the k-horizon if the following conditions hold:*

1. $pstep_k^{(p)}(i_r, i_s) = pstep_k^{(p)}(j_r, j_s)\ \forall r, s \in \{1 \dots n\}$;
2. $pstep_{k-p}^{(p)}(0, i_r) = pstep_{k-p}^{(p)}(0, j_r)\ \forall r \in \{1 \dots n\}$;
3. $pstep_{k-p}^{(p)}(i_r, |w| + 1) = pstep_{k-p}^{(p)}(j_r, |w'| + 1)\ \forall r \in \{1 \dots n\}$.

Local moves are defined on the basis of *pstep-regions*.

Definition 3. *Let $q > 0$, $w \in \Sigma^\star$, and let $\boldsymbol{i}^n$ be a set of positions in w. The* pstep-region $Pstepreg_q^p(w, \boldsymbol{i}^n)$ *is the set* $\{j | \delta(0, j) \leq p \cdot (2^{q-1} - 1) \vee\ \delta(j, |w| + 1) \leq p \cdot (2^{q-1} - 1) \vee\ \exists r (1 \leq r \leq n \wedge \delta(i_r, j) \leq p \cdot 2^{q-1})\}$.

Definition 4. *Let $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ be a configuration and let $q > 0$ be the number of remaining moves. A move is* local *if it is played within $Pstepreg_q^p(w, \boldsymbol{i}^n)$ or $Pstepreg_q^p(w', \boldsymbol{j}^n)$.*

Lemma 1 generalizes the condition of local safety for LSSs [4] by substituting intervals of length p for single positions. If two positions in a string are sufficiently close to each other, the corresponding positions in the other string must be at the same "*pstep* distance". In the following example, we provide a configuration which is not *pstep-safe* in the $(p \cdot 2^3)$-horizon. Moreover, we outline a strategy **I** can adopt to win in 3 rounds.

Example 1. Consider the configuration in Fig. 1. It is immediate to verify that is not *pstep-safe* in the $(p \cdot 2^3)$-horizon. We show a strategy **I** can adopt to win in 3 rounds.

(a) $q = 3$: $pstep_{8p}^{(p)}(i_1, i_2) = 7$ and $pstep_{8p}^{(p)}(j_1, j_2) = 8$. Hence, the configuration is not *pstep-safe* in the $(p \cdot 2^3)$-horizon and thus $\mathbf{I}(\mathcal{G}_3((w, \mathbf{i}^2), (w', \mathbf{j}^2)))$. An optimal move for **I** is to choose $i_3 = i_1 + \lfloor \frac{7}{2} \rfloor \cdot p = i_1 + 3p$ in w. An optimal reply from **II** is to choose $j_3 = j_1 + 3p$ in w'.

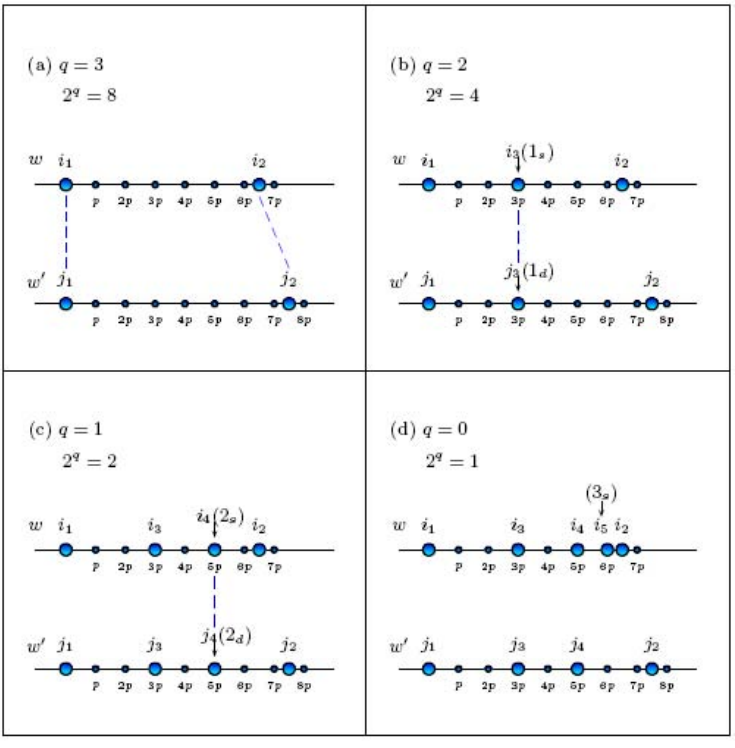


Fig. 1. *pstep*-safety

(b) $q = 2$: $pstep^{(p)}_{4p}(i_3, i_2) = 4$ and $pstep^{(p)}_{4p}(j_3, j_2) = \infty$. Again, the configuration is not *pstep-safe* in the $(p \cdot 2^2)$-horizon and thus $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^3), (w', \mathbf{j}^3)))$. If **I** chooses $i_4 = i_3 + \frac{4}{2} \cdot p = i_3 + 2p$ in w, **II** can reply by choosing $j_4 = j_3 + 2p$ in w'.

(c) $q = 1$: $pstep^{(p)}_{2p}(i_4, i_2) = 2$ and $pstep^{(p)}_{2p}(j_4, j_2) = \infty$. Once more, the configuration is not *pstep-safe* in the $(p \cdot 2^1)$-horizon and thus $\mathbf{I}(\mathcal{G}_1((w, \mathbf{i}^4), (w', \mathbf{j}^4)))$. If **I** chooses $i_5 = i_4 + p$ in w, it holds that $i_4 <_p i_5$ and $i_5 <_p i_2$. Hence, **II** is not able to find an element j_5 such that $j_4 <_p j_5$ and $j_5 <_p j_2$.

(d) $q = 0$: whatever **II**'s move at the previous round has been, **I** is the winner.

Lemma 1. *Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not* pstep-safe *in the $(p \cdot 2^q)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

The next lemma shows that if two positions in a string are sufficiently close to each other, that is, at a distance less than or equal to $2^q - 1$, the corresponding positions in the other string must exactly at the same distance (as a matter of fact, such a property holds for $<$ as well).

As a preliminary step, for every $k > 0$, we introduce a function ϑ_k that computes the truncated signed distance between two positions.

Definition 5. *Let $i, j, k \in \mathbb{N}$, with $i, j, k > 0$. We define the function $\vartheta_k(i, j)$ as follows:*

$$\vartheta_k(i,j) = \begin{cases} i-j & \text{if } \delta_k(i,j) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

Definition 6. *A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* ϑ-safe *in the k-horizon if the following conditions hold:*

1. $\vartheta_k(i_r, i_s) = \vartheta_k(j_r, j_s)\ \forall r, s \in \{1 \dots n\}$;
2. $\vartheta_k(0, i_r) = \vartheta_k(0, j_r)\ \forall r \in \{1 \dots n\}$;
3. $\vartheta_k(i_r, |w| + 1) = \vartheta_k(j_r, |w'| + 1)\ \forall r \in \{1 \dots n\}$.

Lemma 2. *Let $w, w' \in \Sigma^\star$ and $q > 0$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not* ϑ-safe *in the $(2^q - 1)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Observe that if we replace $<_p$ with s, that is, if we take $p = 1$, Lemma 2 becomes a special case of Lemma 1.

Lemma 1 and Lemma 2 basically capture the features that $<_p$ has in common with s and $<$, respectively. However, $<_p$ cannot simply be viewed as the composition of s and $<$. The distinctive features of $<_p$, that differentiate it from s and $<$, are captured by the following definition and will be taken into consideration by Theorem 1.

Let us now introduce the key notions of "rigid" and "elastic" intervals induced by the set of positions $\mathbf{i}^n$ (resp., $\mathbf{j}^n$).

Definition 7. *Let $q > 1$ and $i \in \mathbb{N}$. The 0th q-rigid interval induced by position i is the closed interval $\rho^+_{0,q}(i) = \rho^-_{0,q}(i) = [i - \alpha^0_q, i + \alpha^0_q]$, where $\alpha^0_q = 2^{q-1} - 1$. The kth* right (resp., left) q-rigid interval induced by position i, *with $0 < k \leq 2^{q-2}$, is the interval $\rho^+_{k,q}(i) = (c - \alpha^z_q, c + \alpha^z_q]$ (resp., $\rho^-_{k,q}(i) = [c - \alpha^z_q, c + \alpha^z_q)$) where $c = i + kp$ (resp., $c = i - kp$) and $\alpha^z_q = 1 + \sum_{j=z-1}^{q-2}(2^j - 1) = 2^{q-1} - 2^{z-1} - q + z + 1$, where $z = \lceil \log_2 k \rceil + 1$. The kth* right (resp., left) q-elastic interval induced by position i *is the interval between the $(k-1)$th and the kth q-rigid right (resp., left) interval.*

It is worth noting that elastic intervals come into play only when p is large enough with respect to q. More precisely, in a game $\mathcal{G}_q$, for all $1 \leq k \leq 2^{q-2}$, the kth (right or left) q-elastic interval induced by any position in $\mathbf{i}^n$ is not empty if (and only if) $p > \alpha^z_q + \alpha^{\overline{z}}_q - 1$, where $z = \lceil \log_2(k-1) \rceil + 1$ for $k > 1$ and $z = 0$ for $k = 1$, and $\overline{z} = \lceil \log_2 k \rceil + 1$. In Fig. 2 we show the right 5-rigid (resp., 5-elastic) intervals induced by i, which are represented in black (resp., gray).

The role of rigid and elastic intervals in the evolution of a game is expressed by Theorem 1. It can be intuitively explained as follows. Consider a game $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$, with $q > 1$. If **I** chooses a new position i_{n+1} inside a rigid interval induced by the position $i_r \in \mathbf{i}^n$, that is, if $|i_{n+1} - i_r| \in (kp - \alpha^z_q, kp + \alpha^z_q]$ for a suitable k, then **II** must choose a new position j_{n+1} such that $j_{n+1} - j_r = i_{n+1} - i_r$. If **I** chooses a new position i_{n+1} inside an elastic interval induced by i_r, then **II** must reply by choosing a new position inside the corresponding elastic interval induced by j_r, but not necessarily at the same distance.

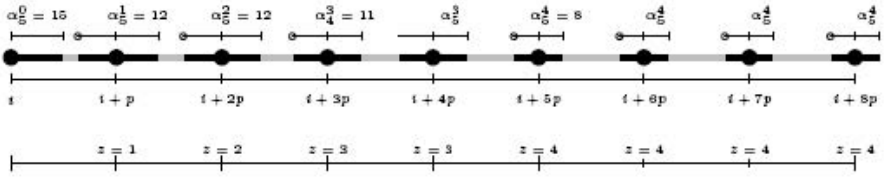


Fig. 2. Rigid and elastic intervals ($q = 5$; $2^{q-2} = 8$)

Definition 8. *Let $q > 0$. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is p-int-*safe *in the k-horizon if the following conditions hold:*

1. *$\forall r, s \in \{1, \ldots, n\}$, with $r < s$, if there exists $0 \leq h \leq 2^{k-1}$ such that $i_s \in \rho^+_{h,k+1}(i_r)$ or $j_s \in \rho^+_{h,k+1}(j_r)$, then $i_s - i_r = j_s - j_r$;*
2. *$\forall r \in \{1, \ldots, n\}$, if there exists $0 \leq h \leq 2^{k-1}$ such that $\delta(0, i_r) \in [hp+1, hp+\alpha^z_{k+1}]$ or $\delta(0, j_r) \in [hp+1, hp+\alpha^z_{k+1}]$, where $z = \lceil \log_2 h \rceil + 1$ for $h > 0$ and $z = 0$ for $h = 0$, then $\delta(0, i_r) = \delta(0, j_r)$;*
3. *$\forall r \in \{1, \ldots, n\}$, if there exists $0 \leq h \leq 2^{k-1}$ such that $\delta(i_r, |w| + 1) \in [hp+1, hp+\alpha^z_{k+1}]$ or $\delta(j_r, |w'|+1) \in [hp+1, hp+\alpha^z_{k+1}]$, where $z = \lceil \log_2 h \rceil + 1$ for $h > 0$ and $z = 0$ for $h = 0$, then $\delta(i_r, |w| + 1) = \delta(j_r, |w'| + 1)$.*

Lemma 3. *Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is p-int-*safe *in the q-horizon, with $q > 0$, then it is ϑ-*safe *in the $(2^q - 1)$-horizon.*

Lemma 4. *Let $w, w' \in \Sigma^\star$ and $q > 0$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not p-int-*safe *in the q-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Proof. The proof is by induction on q.
Base case: $q = 1$. We distinguish three sets of cases:

1. (a) $\exists r, s$ such that $\delta(i_r, i_s) = 1$, but $\delta(j_r, j_s) \neq 1$ (or $\delta(i_r, i_s) \neq 1$, but $\delta(j_r, j_s) = 1$).
   (b) $\exists r, s$ such that $\delta(i_r, i_s) = p$, but $\delta(j_r, j_s) \neq p$ (or $\delta(i_r, i_s) \neq p$, but $\delta(j_r, j_s) = p$).
   (c) $\exists r, s$ such that $\delta(i_r, i_s) = p+1$, but $\delta(j_r, j_s) \neq p+1$ (or $\delta(i_r, i_s) \neq p+1$, but $\delta(j_r, j_s) = p + 1$).
2. (a) $\exists r$ such that $\delta(0, i_r) = 1$, but $\delta(0, j_r) \neq 1$ (or $\delta(0, i_r) \neq 1$, but $\delta(0, j_r) = 1$).
   (b) $\exists r$ such that $\delta(0, i_r) = p + 1$, but $\delta(0, j_r) \neq p + 1$ (or $\delta(0, i_r) \neq p + 1$, but $\delta(0, j_r) = p + 1$).
3. (a) $\exists r$ such that $\delta(i_r, |w|+1) = 1$, but $\delta(j_r, |w'|+1) \neq 1$ (or $\delta(i_r, |w|+1) \neq 1$, but $\delta(j_r, |w'| + 1) = 1$).
   (b) $\exists r$ such that $\delta(i_r, |w| + 1) = p + 1$, but $\delta(j_r, |w'| + 1) \neq p + 1$ (or $\delta(i_r, |w| + 1) \neq p + 1$, but $\delta(j_r, |w'| + 1) = p + 1$).

We provide the details for the first set of cases; the others are similar and thus omitted.

(a) The configuration is not ϑ-*safe* in the (2^1-1)-horizon and thus, by Lemma 2, $\mathbf{I}(\mathcal{G}_1((w,\mathbf{i}^n),(w',\mathbf{j}^n)))$.
(b) If $\delta(j_r,j_s)>p$, then the configuration is not a partial isomorphism. Hence, we restricted our attention to the case in which $\delta(j_r,j_s)<p$. Furthermore, we assume that i_r-i_s and j_r-j_s have the same sign (if this is not the case, the configuration is not *pstep-safe* in the $(p\cdot 2^0)$-horizon, and thus it is not a partial isomorphism). Without loss of generality, let $i_r<i_s$ and $j_r<j_s$. An optimal move for **I** is to choose $j_{n+1}=j_s-p$. If **II** chooses $i_{n+1}<i_r$, then $i_s-i_{n+1}>p$; if **II** chooses $i_{n+1}>i_r$, then $j_{n+1}<j_r$ but $i_{n+1}>i_r$.
(c) If $\delta(j_r,j_s)\leq p$, the configuration is not a partial isomorphism. Hence, let $\delta(j_r,j_s)>p+1$. Furthermore, let i_r-i_s and j_r-j_s have the same sign. Without loss of generality, let $i_r<i_s$ and $j_r<j_s$. An optimal move for **I** is to choose $j_{n+1}=j_r+p+1(<j_s)$. If **II** replies with $i_{n+1}\geq i_s$, the configuration is not a partial isomorphism; if **II** replies with $i_{n+1}<i_s$, then $\delta(i_r,i_{n+1})\leq p$ while $\delta(j_r,j_{n+1})=p+1$ and thus the configuration is not a partial isomorphism.

Inductive step: $q>1$. As in the base case, we distinguish three sets of cases:

1. (a) $\exists r,s\in\{1,\ldots,n\}$ such that $\delta(i_r,i_s)\leq 2^q-1$ or $\delta(j_r,j_s)\leq 2^q-1$ but $i_r-i_s\neq j_r-j_s$.
   (b) $\exists r,s\in\{1,\ldots,n\}$, $\exists 1\leq k\leq 2^{q-1}$ such that $i_s\in\rho^+_{k,q+1}(i_r)$ or $j_s\in\rho^+_{k,q+1}(j_r)$ but $i_r-i_s\neq j_r-j_s$.
2. (a) $\exists r\in\{1,\ldots,n\}$ such that $\delta(0,i_r)\leq 2^q-1$ or $\delta(0,j_r)\leq 2^q-1$, but $\delta(0,i_r)\neq\delta(0,j_r)$.
   (b) $\exists r\in\{1,\ldots,n\}$, $\exists 1\leq k\leq 2^{q-1}$ such that $\delta(0,i_r)\in[kp+1,kp+\alpha^z_{q+1}]$ or $\delta(0,j_r)\in[kp+1,kp+\alpha^z_{q+1}]$, where $z=\lceil\log_2 k\rceil+1$, but $\delta(0,i_r)\neq\delta(0,j_r)$.
3. (a) $\exists r\in\{1,\ldots,n\}$ such that $\delta(i_r,|w|+1)\leq 2^q-1$ or $\delta(j_r,|w'|+1)\leq 2^q-1$, but $\delta(i_r,|w|+1)\neq\delta(j_r,|w'|+1)$.
   (b) $\exists r\in\{1,\ldots,n\}$, $\exists 1\leq k\leq 2^{q-1}$ such that $\delta(i_r,|w|+1)\in[kp+1,kp+\alpha^z_{q+1}]$ or $\delta(j_r,|w'|+1)\in[kp+1,kp+\alpha^z_{q+1}]$, where $z=\lceil\log_2 k\rceil+1$, but $\delta(i_r,|w|+1)\neq\delta(j_r,|w'|+1)$.

As in the base case, we provide the details for the first set of cases; the others are similar and thus omitted.

(a) Since the configuration is not ϑ-*safe* in the (2^q-1)-horizon, by Lemma 2 $\mathbf{I}(\mathcal{G}_1((w,\mathbf{i}^n),(w',\mathbf{j}^n)))$.
(b) Let us suppose that $i_s\in\rho^+_{k,q+1}(i_r)$. We partition the rigid interval $\rho^+_{k,q+1}(i_r)$ into five parts and we follow a different strategy for each of them. Let c be the center of $\rho^+_{k,q+1}(i_r)$ and let α^z_{q+1}, where $z=\lceil\log k\rceil+1$, be its radius. Without loss of generality, we assume that $i_r<i_s$ and $j_r<j_s$. From left to right, the five subintervals of $\rho^+_{k,q+1}(i_r)$ we are going to consider are the following:

1. $(c-\alpha^z_{q+1},c-2^{q-1}]$
2. $[c-2^{q-1}+1,c]$
3. $(c,c+2^{q-1}-1]$
4. $[c+2^{q-1},c+\alpha^z_{q+1})$
5. $c+\alpha^z_{q+1}$ (the right endpoint of $\rho^+_{k,q+1}(i_r)$)

Strategy 1: Let $\delta(i_r, i_s) = kp - s$, with $s \in [2^{q-1}, \alpha^z_{q+1})$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r, i_s) > \delta(j_r, j_s) \geq kp - (p+1)$ (if the last condition is not satisfied, *pstep-safety* is violated and the thesis immediately follows). **I** chooses $j_{n+1} = j_r - (2^{q-1} - 1)$. If **II** replies with $i_{n+1} \neq i_r - (2^{q-1} - 1)$, the configuration is not ϑ-*safe* in the 2^{q-1}-horizon, then $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with $i_{n+1} = i_r - (2^{q-1} - 1)$, then $(kp >)\delta(i_{n+1}, i_s) = kp - s + (2^{q-1} - 1) \geq kp - \alpha^z_{q+1} + 1 + (2^{q-1} - 1) = kp - \alpha^z_q + 1$. From $\delta(i_{n+1}, i_s) > \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 2: Let $\delta(i_r, i_s) = kp - s$, with $s \in [0, 2^{q-1} - 1]$, and let $\delta(i_r, i_s) > \delta(j_r, j_s) \geq kp - (p+1)$. **I** chooses $j_{n+1} = j_s - kp$. If **II** replies with $i_{n+1} = i_s - kp$, then $0 \leq \delta(i_{n+1}, i_r) \leq 2^{q-1} - 1$ and $\delta(j_{n+1}, j_r) > \delta(i_{n+1}, i_r)$. It immediately follows that the configuration is not ϑ-*safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with i_{n+1} such that $\delta(i_{n+1}, i_r) = \delta(j_{n+1}, j_r)$, then $pstep_{p \cdot 2^{q-1}}(i_{n+1}, i_s) \neq pstep_{p \cdot 2^{q-1}}(j_{n+1}, j_s)$. As a consequence, the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 3: Let $\delta(i_r, i_s) = kp + s$, with $s \in (0, 2^{q-1} - 1]$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $i_{n+1} = i_s - kp$. If **II** replies with $j_{n+1} = j_s - kp$, then $0 < \delta(i_r, i_{n+1}) \leq 2^{q-1} - 1$. From $\delta(j_r, j_{n+1}) > \delta(i_r, i_{n+1})$, it follows that the configuration is not ϑ-*safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with j_{n+1} such that $\delta(i_r, i_{n+1}) = \delta(j_r, j_{n+1})$, then $pstep_{p \cdot 2^{q-1}}(i_{n+1}, i_s) \neq pstep_{p \cdot 2^{q-1}}(j_{n+1}, j_s)$. As a consequence, the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 4: Let $\delta(i_r, i_s) = kp + s$, with $s \in [2^{q-1}, \alpha^z_{q+1})$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $i_{n+1} = i_r + 2^{q-1} - 1$. If **II** replies with $j_{n+1} \neq j_q + 2^{q-1} - 1$, the configuration is not ϑ-*safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with $j_{n+1} = j_r + 2^{q-1} - 1$, then $(kp <)\delta(i_{n+1}, i_s) = kp + s - (2^{q-1} - 1) \leq kp + \alpha^z_{q+1} - (2^{q-1} - 1) = kp + \alpha^z_q$. From $\delta(i_{n+1}, i_s) < \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 5: Let $\delta(i_r, i_s) = kp + \alpha^z_{q+1}$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $j_{n+1} = j_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, where $z' = \lceil \log_2 \lceil k/2 \rceil \rceil + 1$. By definition, $j_{n+1} \in \rho^+_{\lceil k/2 \rceil, q}(j_r)$. If **II** replies with $i_{n+1} \neq i_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with $i_{n+1} = i_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, it is possible to show that $i_{n+1} \in \rho^-_{\lfloor k/2 \rfloor, q}(i_s)$. From $\delta(i_{n+1}, i_s) \neq \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Definition 9. *A configuration* $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ *is* q-distance-safe *if it is pstep*-safe *in the* $(p \cdot 2^q)$*-horizon and, if* $q > 0$*, it is p-int*-safe *in the* q*-horizon.*

The next theorem takes advantage of previous lemmas to provide a necessary condition for **II** to win.

Theorem 1. *[Necessary condition for **II** to win]*
*Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not q-*distance-safe*, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Proof. If $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not *pstep-safe* in the $(p \cdot 2^q)$-horizon, then, by Lemma 1, $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$; if $q > 0$ and $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not *p-int-safe* in the q-horizon, then, by Lemma 4, $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$.

The next theorem states that *q-distance-safety* is also a sufficient condition for **II** to win (in the restricted setting we are considering, where only local moves are allowed).

Theorem 2. *[Sufficient condition for **II** to win]*
Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is q-distance-safe, then $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

4 Local Games on Labeled $<_p$ Structures

In this section, we consider the effects of adding unary predicates, that is, of associating a label with each string position, to the local games studied in the previous section. To this end, we introduce the notions of *q-color of a position* and of *q-color of an interval*, which gives a recursive characterization of the labels occurring in a suitable neighborhood of the position.

Definition 10. *Let $i, j, p \in \mathbb{N}$, with $i, j, p \geq 1$. We define $p\text{-}int_j^+(i) = [i + (j - 1)p + 1, i + jp]$ and $p\text{-}int_j^-(i) = [i - jp, i - (j - 1)p - 1]$.*

In the following definition, the q-color of positions is defined in terms of the $(q-1)$-color of intervals, which in its turn is defined in terms of the $(q-1)$-color of positions. Moreover, we distinguish between the q-color of an internal interval and the q-color of prefix and suffix intervals. Finally, to keep definitions as simple as possible, we assume all (fictitious) positions before position 1 and after position $|w|$ to be labeled by the special symbol \$.

Definition 11. *Let $w \in \Sigma^*$, $q, p \in \mathbb{N}$, with $p > 1$, and $i \in \mathbb{Z}$. The* q-color of position i in w, *denoted by $q\text{-col}_w(i)$, is inductively defined as follows:*

- *the 0-color of i in w is the label $w[i]$ for $i \in \{1 \dots |w|\}$ and \$ otherwise;*
- *the (q+1)-color of i in w is the ordered tuple $\sigma_{2^q}^w \cdots \sigma_1^w w[i] \tau_1^w \dots \tau_{2^q}^w$ where for all $1 \leq j \leq 2^q$, τ_j^w is the q-color of $p\text{-}int_j^+(i)$ and σ_j^w is the q-color of $p\text{-}int_j^-(i)$.*

The q-color *of the jth right (resp., left) interval $[a, b] = p\text{-}int_j^+(i)$ (resp., $p\text{-}int_j^-(i)$) induced by i, abbreviated q-col-right-int$_w^j(a, b)$ (resp., q-col-left-int$_w^j(a, b)$), with $1 \leq j \leq 2^q$, is the ordered tuple $t_a^w \dots t_{a+\gamma_1-1}^w \{t_{a+\gamma_1}^w \dots t_{b-\gamma_2}^w\} t_{b-\gamma_2+1}^w \dots t_b^w$ (resp., $t_a^w \dots t_{a+\gamma_2-1}^w \{t_{a+\gamma_2}^w \dots t_{b-\gamma_1}^w\} t_{b-\gamma_1+1}^w \dots t_b^w$), where for all $a \leq k \leq b$, $t_k^w = q\text{-}col_w(k)$ and*

$$\gamma_1 = \begin{cases} \alpha_{q+1}^z & \text{if } q \neq 0 \text{ and } j - 1 \leq 2^{q-1}, \text{ where } z = \lceil \log_2(j-1) \rceil + 1 \text{ for } j > 1 \\ \quad \text{and } z = 0 \text{ for } j = 1, & \\ 0 & \text{if } q = 0 \text{ or } j - 1 > 2^{q-1}. \end{cases}$$

$$\gamma_2 = \begin{cases} \alpha_{q+1}^z - 1 & \text{if } q \neq 0 \text{ and } j \leq 2^{q-1}, \text{ where } z = \lceil \log_2 j \rceil + 1 \\ 0 & \text{if } q = 0 \text{ or } j > 2^{q-1}. \end{cases}$$

For $q > 0$, the q-color *of the jth prefix (resp., suffix) interval $[a, b] = p\text{-}int_j^+(0)$ (resp., $p\text{-}int_j^-(|w| + 1)$), abbreviated* q-col-pref$_w^j(a, b)$ *(resp.,* q-col-suff$_w^j(a, b)$*), with $1 \leq j \leq 2^q - 1$, $q > 0$, is the ordered tuple $t_a^w \dots t_{a+\gamma_1-1}^w \{t_{a+\gamma_1}^w \dots t_b\}$ (resp., $\{t_a^w \dots t_{b-\gamma_1}^w\} t_{b-\gamma_1+1}^w \dots t_b$), where $\forall a \leq i \leq b$, $t_i^w = q\text{-}col_w(i)$ and*

$$\gamma_1 = \begin{cases} \alpha_{q+1}^z & \text{if } q \neq 0 \text{ and } j - 1 \leq 2^{q-1}, \text{ where } z = \lceil \log_2(j-1) \rceil + 1 \text{ for } j > 1 \\ \quad \text{and } z = 0 \text{ for } j = 1, & \\ 0 & \text{if } j - 1 > 2^{q-1}. \end{cases}$$

The next definition introduces the notion of $<_p$*-safety for q-colors.*

Definition 12. *Let $w, w' \in \Sigma^*$ and $p, n, q \in \mathbb{N}$, with $p > 0$. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* $<_p$-safe for q-colors *if the following conditions hold:*

- $\forall r \in \{1, \dots, n\}$, $q\text{-}col_w(i_r) = q\text{-}col_{w'}(j_r)$;
- $\forall 1 \leq j \leq 2^{q-1} - 1$, *with* $q > 1$, *$(q-1)$-col-pref$_w^j(p\text{-}int_j^+(0)) = (q-1)$-col-pref$_{w'}^j(p\text{-}int_j^+(0))$;*
- $\forall 1 \leq j \leq 2^{q-1} - 1$, *with* $q > 1$, *$(q-1)$-col-suff$_w^j(p\text{-}int_j^-(|w| + 1)) = (q-1)$-col-suff$_{w'}^j(p\text{-}int_j^-(|w'| + 1))$.*

Example 2. In Figs. 3 and 4 we show two configurations which are not $<_p$*-safe for 2-colors*. Moreover, for each of them, we outline a strategy **I** can adopt to win in 2 rounds.

As for Fig. 3, a condition relative to the prefix of an interval is violated. We have that $2\text{-}col_2^w(i_1) = \sigma_2^w \sigma_1^w w[i_1] \tau_1^w \tau_2^w$ and $2\text{-}col_2^{w'}(j_1) = \sigma_2^{w'} \sigma_1^{w'} w'[j_1] \tau_1^{w'} \tau_2^{w'}$. If we consider only the portions of w and w' on the right of i_1 and j_1, we have that $\tau_1^w = \dots..a\{a,b\}\{..b\{a,b\}\}$, $\tau_1^{w'} = \dots..a\{a,b\}\{..b\{a,b\}\}$, $\tau_2^w = \dots..a\{a,b\}\{\dots\}$, and $\tau_2^{w'} = \dots..b\{a,b\}\{\dots\}$. Since $1\text{-}col_w(i_1 + p + 1) \neq 1\text{-}col_{w'}(j_1 + p + 1)$, it holds that $2\text{-}col_w(i_1) \neq 2\text{-}col_{w'}(j_1)$. Then $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^1), (w', \mathbf{j}^1)))$. An optimal move for **I** is to choose $i_2 = i_1 + p + 1$ in w. **II** must choose a position within $[j_1+p+1, \dots, j_1+2p]$ in w' labeled by a. Let us suppose **II** chooses $j_2 = j_1+p+2$. The new configuration is not *p-int-safe* in the 1-horizon. If **I** chooses $j_3 = j_1+1$, it holds that $j_3 \not<_p j_2$. Hence, **II** is not able to find a position i_3 in w such that $i_3 > i_1$ and $i_3 \not<_p i_2$.

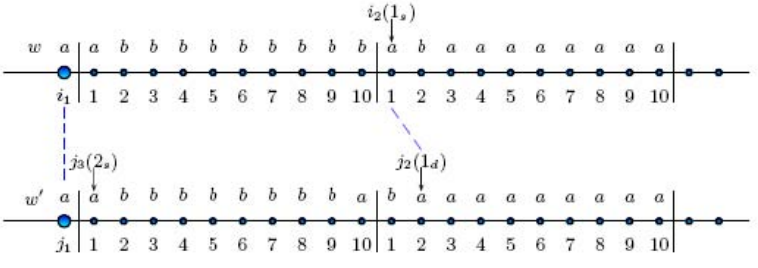


Fig. 3. Safety for q-colors ($q = 2$, $\Sigma = \{a, b\}$, $p = 10$)

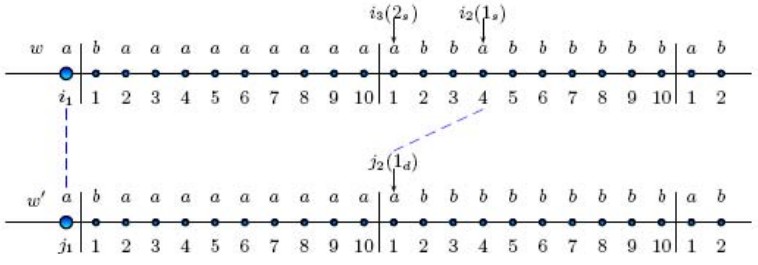


Fig. 4. Safety for q-colors ($q = 2$, $\Sigma = \{a, b\}$, $p = 10$)

As for Fig. 4, a condition relative to the interior of an interval is violated. We have that $2\text{-}col_2^w(i_1) = \sigma_2^w \sigma_1^w w[i_1] \tau_1^w \tau_2^w$ and $2\text{-}col_2^{w'}(j_1) = \sigma_2^{w'} \sigma_1^{w'} w'[j_1] \tau_1^{w'} \tau_2^{w'}$. If we consider only the portions of w and w' on the right of i_1 and j_1, we have that $\tau_1^w = \ldots..b\{a\}\{..a\{a,b\}\}$, $\tau_1^{w'} = \ldots..b\{a\}\{..a\{a,b\}\}$, $\tau_2^w = \ldots..a\{a,b\}\{..b\{a,b\}$, $..a\{a,b\}\}$, and $\tau_2^{w'} = \ldots..a\{a,b\}\{..b\{a,b\}\}$. Since $\tau_2^w \neq \tau_2^{w'}$ (in particular, $\{b\{a,b\},\ a\{a,b\}\} \neq \{b\{a,b\}\}$), it holds that $2\text{-}col_w(i_1) \neq 2\text{-}col_{w'}(j_1)$. Then $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^1),\ (w', \mathbf{j}^1)))$. An optimal move for **I** is to choose $i_2 = i_1 + p + 4$, where $w[i_1 + p + 4] = a$. **II** must choose a position within $[j_1 + p + 1, \ldots, j_1 + 2p]$ labeled by a. He can only choose $j_2 = j_1 + p + 1$. The new configuration is not *p-int-safe* in the 1-horizon. If **I** chooses $i_3 = i_1 + p + 1$, it holds that $i_1 \not<_p i_3$. Hence, **II** is not able to find a position j_3 in w' such that $j_1 < j_3 < j_2$ and $j_1 \not<_p j_3$.

The following theorem provides a necessary condition for **II** to win a local game on labeled $<_p$ structures.

Theorem 3. *[Necessary condition for **II** to win]*
Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p > 1$. If $(w, w', \boldsymbol{i}^n,\ \boldsymbol{j}^n)$ is not $<_p$-safe for q-colors, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

The next theorem gives a sufficient condition for **II** to win a local game on labeled $<_p$ structures. It pairs the condition of *q-distance-safety* (Theorem 2) with that of *$<_p$-safety for q-colors*.

Definition 13. *A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* q-locally-safe *if it is* q-distance-safe *in the $(p \cdot 2^q)$-horizon and* $<_p$-safe for q-colors.

Theorem 4. *[Sufficient condition for **II** to win]*
Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p > 1$. If $(w, w', \boldsymbol{i}^n,\ \boldsymbol{j}^n)$ is q-locally-safe, then $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n),\ (w',\ \boldsymbol{j}^n)))$.

To summarize, we have that *q-local safety* and *$<_p$-safety for q-colors* are necessary and sufficient conditions for **II** to have a winning strategy in q rounds when **I**'s moves are coerced to be local.

We conclude the section by showing that it is possible to check in polynomial time whether the q-colors of two positions in the two strings are equal or not.

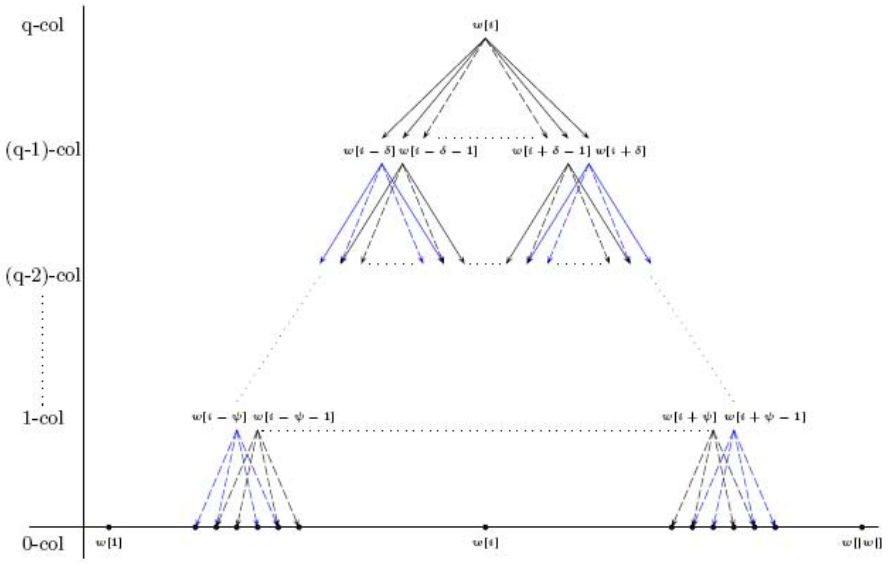


Fig. 5. The proposed representation of $q\text{-col}_w(i)$

We take advantage of a suitable representation of q-colors, which is graphically depicted in Fig. 5. The x-axis refers to string positions, while the y-axis refers to q-colors. The q-color of a position i is represented by the label $w[i]$ plus a set of pointers (arrows) to the relevant $(q-1)$-colors, that is, to the $(q-1)$-colors of positions $i-\delta,\ldots,i-1,i+1,\ldots,i+\delta$, with $\delta = p\cdot 2^{q-1}$. Labeled positions and pointers give rise to a directed layered graph. The number of nodes and edges of such a graph can be computed as follows. Let $|w| = n$. For $0 \le k < q$, the number of nodes of the graph at level k equals the number of k-colors to be computed, that is, $2\cdot p\cdot\sum_{l=k}^{q-1} 2^l = O(n)$. Thus, the total number of nodes is $O(n\cdot q)$. As far as the number of edges is concerned, for $1 \le k \le q$, at level k the number of outgoing edges is $(2\cdot p\sum_{l=k}^{q-1} 2^l)\cdot(2\cdot p\cdot 2^{k-1}) = O(n^2)$. Hence, the total number of edges is $O(n^2\cdot q)$. We partition the set of edges in two classes: *continuous* edges, that point to colors that must occur at the same position within the corresponding p-intervals (of the two strings) and *dashed* edges, that point to colors whose position within the corresponding p-intervals (of the two strings) is irrelevant. To compare the q-colors of two given positions (in the two strings), we build the corresponding graphs and we visit them in a bottom-up fashion, that is, we start from nodes representing 0-colors and we move upwards level-by-level. For both graphs, we compute, at each level k, with $0 \le k < q$, a table that, for every k-color, keeps track of the its occurrences in the other graph. More precisely, for each position $1 \le i \le 2\cdot p\cdot\sum_{l=k}^{q-1} 2^l$, we compute all the occurrences of *k-col(i)* in the other graph. As a matter of fact, it suffices to search for the occurrences of *k-col(i)* in the interval $[i-(q-k)\cdot(p-1),\ldots,i+(q-k)\cdot(p-1)]$. Moreover, to build the table at level k, we only need to look at the table at level $k-1$. Hence, once the former has been computed, the latter can be deleted. The table at level

0 can be easily computed by looking for the occurrences of each 0-color in the other string. To build the table at level k, with $0 < k < q$, we need to compare k-colors in the two graphs. More precisely, for every position we must check the presence of every $(k-1)$-color it points to in the corresponding p-interval of the other string. In fact, if a $(k-1)$-color is reached by a continuous arrow, it suffices to check whether it occurs exactly at the same position within the corresponding p-interval (in the other string). This can be done by accessing the table at level $k-1$. Once the first "k-comparison" has been done, the following ones can exploit the outcomes of the "$(k-1)$-searches" already performed, because the computation of the k-colors of two neighbor positions requires the computation of the $(k-1)$-colors of two portions of the string which only differ for a 1-position shift.

The resulting algorithm has the following complexity. To build the table at level k, we compare each of the $O(n)$ k-colors of a string with each of the $O(n)$ k-colors in the other string. The comparison of two k-colors takes time $O(p^2 2^k)(= O(p^2 n))$, because it requires to take into consideration $p2^k$ positions, that is, entries of the table at level k-1, and to scan (part of) their occurrence list of length $O(p)$. Thus, the table at level k can be built in time $O(p^2 n^3)$. Since we need to build q-1 tables, the overall time complexity is $O(p^2 n^3 q)$. Given that q is logarithmic in n (it is not difficult to show that if **I** has a winning strategy, then he has a winning strategy in $O(\log n)$ rounds), the complexity of the algorithm is $O(p^2 n^3 \log n)$, which is polynomial in the length of the strings.

5 Global Games on Labeled $<_p$ Structures

In this section, we consider the general case, where we do not constrain **I**'s moves to be local. The solution consists in combining a local and a global strategy. The global strategy essentially requires to compare the *multiplicity* and *scattering* of r-colors τ in the portions of w and w' that are far away from already selected positions.

Definition 14. *Let $q, p \in \mathbb{N}^+$, $\boldsymbol{i}^n$ be a set of positions in w and τ be a $(q-1)$-color. The* (q,p)-free-multiplicity *of τ in w, abbreviated $\rho^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau)$, is the number of occurrences of τ in w which fall in $Free^p_q(w, \boldsymbol{i}^n)$, where $Free^p_q(w, \boldsymbol{i}^n) = \{1, \ldots, |w|\} \setminus Pstepreg^p_q(w, \boldsymbol{i}^n)$.*

Definition 15. *Let $P \subseteq \mathbb{N}$ be a finite set. A* k-blurred partition $\mathcal{P}$ *of P is a partition of P such that (i) for each $A \in \mathcal{P}$ and for each $a, b \in A$, $\delta(a,b) \leq k$, and (ii) there is not a partition $\mathcal{P}'$ satisfying (i) such that $|\mathcal{P}| > |\mathcal{P}'|$. The number of classes of $\mathcal{P}$ is called* k-blurring.

Definition 16. *Let $q, p \in \mathbb{N}^+$, $\boldsymbol{i}^n$ be a set of positions in w and τ be a $(q-1)$-color. The* (q,p)-free-scattering *of τ in w, abbreviated $\sigma^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau)$, is the $(p2^q)$-blurring of $\{i \mid (q-1)\text{-}color_w(i) = \tau \wedge i \in Free^p_q(w, \boldsymbol{i}^n)\}$, where $Free^p_q(w, \boldsymbol{i}^n) = \{1, \ldots, |w|\} \setminus Pstepreg^p_q(w, \boldsymbol{i}^n)$.*

Let $\Delta^{(w,\mathbf{i}^n)}_{(w',\mathbf{j}^n)} = \{\tau \mid \tau \text{ is a (q-1)-color}, q > 0, \text{ and } \sigma^{(w,\mathbf{i}^n)}_{(q,p)}(\tau) \neq \sigma^{(w',\mathbf{j}^n)}_{(q,p)}(\tau) \vee \rho^{(w,\mathbf{i}^n)}_{(q,p)}(\tau) \neq \rho^{(w',\mathbf{j}^n)}_{(q,p)}(\tau)\}$ be the set of words that **I** can exploit to win. The next theorem provides a necessary condition for **II** to win.

Theorem 5. *Let $q, p, n \in \mathbb{N}$, with $q, p > 0$, and let $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ be a configuration. If there exists a $q-1$-color τ such that $\tau \in \Delta^{(w,\boldsymbol{i}^n)}_{(w',\boldsymbol{j}^n)}$, then $\boldsymbol{I}(\mathcal{G}_{q+min\{\sigma^{(w,i^n)}_{(q,p)}(\tau),\sigma^{(w',j^n)}_{(q,p)}(\tau)\}}((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

A (necessary and) sufficient condition for **II** to win is given by the following theorem.

Theorem 6. *[Main Theorem]*
Let $w, w' \in \Sigma^*$ and $p, q \in \mathbb{N}$, with $p > 1$. $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$ if and only if the following conditions hold:

1. *$(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is q-locally-safe;*
2. *for all $(r-1)$-color $\tau \in \Delta^{(w,\boldsymbol{i}^n)}_{(w',\boldsymbol{j}^n)}$, with $1 \leq r \leq q$, $\sigma^{(w,\boldsymbol{i}^n)}_{(i,p)}(\tau) > q - r$ and $\sigma^{(w',\boldsymbol{j}^n)}_{(i,p)}(\tau) > q - r$.*

Condition 2. means that **I** cannot detect any difference in q rounds if the *scattering* of the $(r-1)$-colors that have different multiplicity or scattering in $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$ is sufficiently high in both structures, where the thresholds depend on the size of the $(r-1)$-colors (higher thresholds for smaller $(r-1)$-colors). The remoteness of $\mathcal{G}$ is thus $r + \min(\sigma^{(w,\mathbf{i}^n)}_{(r,p)}, \sigma^{(w',\mathbf{j}^n)}_{(r,p)})$ and it can be intuitively explained as follows: given a suitable $\tau \in \Delta^{(w,\mathbf{i}^n)}_{(w',\mathbf{j}^n)}$, **I** can first choose $\min(\sigma^{(w,\mathbf{i}^n)}_{(r,p)}, \sigma^{(w',\mathbf{j}^n)}_{(r,p)})$ occurrences of τ and then play a local move; after that, it is guaranteed that the reached position is not $(r-1)$-locally safe, so **I** can win the game by playing $r-1$ other local moves.

To compute the remoteness of two strings w and w', we can thus proceed as follows. Let $n = min\{|w|, |w'|\}$. As we already pointed out, this gives an upper bound $m = \log n$ to the value of remoteness (unless the two strings coincide). Hence, for $i = 1, \ldots, m$, we search for an $(i-1)$-color with a different scattering or multiplicity in w and w' (if any). As a preliminary step, we construct a layered graph that represents the $(m-1)$-color of all the positions of the string w. Then, we repeat the same construction for the string w'. Both the resulting graphs consists of $O(n \log n)$ nodes and $O(n^2 \log n)$ edges. Next, for $i = 1, \ldots, |w|$ and $k = 0, \ldots, m-1$, we build a table that keeps track of the occurrences of the k-color of position i in w and w' (in fact, we must also deal with the k-colors of the fictitious positions preceding position 1 and following position $|w|$; however, the treatment of such cases does not affect the complexity of the building procedure). Then, we repeat the same construction for w'. Both constructions take time $O(p^2 n^3 \log n)$. The resulting tables can then be exploited to compute multiplicity and scattering of each k-color in time $O(n)$. This last step globally takes time $O(n^2 \log n)$. The overall complexity of the computation is thus $O(p^2 n^3 \log n)$.

6 Conclusions

In this paper, we analyzed *EF-games* on labeled $<_p$ structures. The $<_p$ relation features a mix of the characteristics of the successor relation s and the linear order relation $<$. From s, it borrows the condition of *pstep-safety* (such a condition is trivially satisfied in the case of $<$). Moreover, it shares the condition of *θ-safety* with $<$ (in the case of s, such a condition immediately follows from *pstep-safety*). In addition, it features some distinctive characteristics, such as the partition of the neighborhoods of selected positions in rigid and elastic intervals.

The paper identified necessary and sufficient winning conditions for **I** and **II**, that allow one to compute the remoteness of a game and optimal strategies for both players. Moreover, it provides a polynomial algorithm for the crucial step in the computation of the remoteness, namely, checking whether the q-colors of two distinct positions are equal.

We are currently comparing the values of the remoteness of real biological sequences (in particular, DNA sequences of the plasmodium parasites), as well as of artificial sequences, for different values of the parameter p. We are also studying the case in which, instead of fixing the value of p in advance, we make p a function of the size of the input sequences, e.g., a logarithmic function.

References

1. Arora, S., Fagin, R.: On winning strategies in Ehrenfeucht-Fraïssé games. Theoretical Computer Science 174, 97–121 (1997)
2. Keisler, H.J., Lotfallah, W.B.: Shrinking games and local formulas. Annals of Pure and Applied Logic 128, 215–225 (2004)
3. Schwentick, T.: On winning Ehrenfeucht games and monadic NP. Annals of Pure and Applied Logic 79, 61–92 (1996)
4. Montanari, A., Policriti, A., Vitacolonna, N.: An Algorithmic Account of Winning Strategies in Ehrenfeucht Games on Labeled Successor Structures. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS, vol. 3835, pp. 139–153. Springer, Heidelberg (2005)
5. Pezzoli, E.: Computational Complexity of Ehrenfeucht-Fraïssé Games on Finite Structures. In: Gottlob, G., Grandjean, E., Seyr, K. (eds.) CSL 1998. LNCS, vol. 1584, pp. 159–170. Springer, Heidelberg (1999)
6. Khoussainov, B., Liu, J.: On Complexity of Ehrenfeucht-Fraïssé Games. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 293–309. Springer, Heidelberg (2007)
7. De Maria, E., Montanari, A., Vitacolonna, N.: Games on strings with a limited order relation. Technical Report 11/08, Department of Mathematics and Computer Science, University of Udine, Italy (2008)

Complete Axiomatizations of MSO, FO(TC^1) and FO(LFP^1) on Finite Trees*

Amélie Gheerbrant[1] and Balder ten Cate[2]

[1] ILLC, Universiteit van Amsterdam
a.gheerbrant@uva.nl
[2] ISLA, Universiteit van Amsterdam
balder.tencate@uva.nl

Abstract. We propose axiomatizations of monadic second-order logic (MSO), monadic transitive closure logic (FO(TC^1)) and monadic least fixpoint logic (FO(LFP^1)) on finite node-labeled sibling-ordered trees. We show by a uniform argument, that our axiomatizations are complete, i.e., in each of our logics, every formula which is valid on the class of finite trees is provable using our axioms. We are interested in this class of structures because it allows to represent basic structures of computer science such as XML documents, linguistic parse trees and treebanks. The logics we consider are rich enough to express interesting properties such as reachability. On arbitrary structures, they are well known to be not recursively axiomatizable.

Keywords: Trees, Axiomatizations, Fragments of MSO, Henkin semantics, Ehrenfeucht-Fraïssé games, Feferman-Vaught theorems.

We develop a uniform method for obtaining complete axiomatizations of fragments of MSO on trees. In particular, we obtain a complete axiomatization for MSO, FO(TC^1), and FO(LFP^1) on finite node-labeled sibling-ordered trees. We take inspiration from Kees Doets, who proposed in (4) a complete axiomatization of first-order logic (FO) on the class of node-labeled finite trees without sibling-order. A similar result was shown in (1) and (18) for FO on node-labeled finite trees with sibling order. We use the signature of (18) and extend the set of axioms proposed there.

Finite trees are basic and ubiquitous structures which are of interest at least to mathematicians, computer scientists (tree-structured documents) and linguists (parse trees). The logics we study are known to be very well-behaved on this particular class of structures and to have an interestingly high expressive power.

* We are grateful to Jouko Väänänen for helpful comments on an earlier draft. A full version including proofs is available online http://www.illc.uva.nl/Publications/ResearchReports/PP-2008-44.text.pdf. The authors are supported by a GLoRiClass fellowship of the European Commission (Research Training Fellowship MEST-CT-2005-020841) and by the Netherlands Organization for Scientific Research (NWO) grant 639.021.508, respectively.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 180–196, 2009.

In particular, they all allow to express reachability, but at the same time, they have the advantage of being decidable on trees.

As XML documents are tree-structured data, our results are particularly relevant to XML query languages. Query languages are logical languages used to make queries into database and information systems. In (19) and (8), MSO and FO(TC^1) have been proposed as a yardstick of expressivity on trees for these languages. It is known that FO(LFP^1) has the same expressive power as MSO on trees, but the translations between the two are non-trivial, and hence it is not clear whether an axiomatization for one language can be obtained from an axiomatization for the other language in any straightforward way.

In applications to computational linguistics, finite trees are used to represent the grammatical structure of natural language sentences. In the context of *model theoretic syntax*, Rogers advocates in (17) the use of MSO in order to characterize derivation trees of context free grammars. Kepser also argues in (11) that MSO should be used in order to query treebanks. A treebank is a text corpus in which each sentence has been annotated with its syntactic structure (represented as a tree structure). In (12) and (20) Kepser and Tiede propose to consider various transitive closure logics, among which FO(TC^1), arguing that they constitute very natural formalisms from the logical point of view, allowing concise and intuitive phrasing of parse tree properties.

The remainder of the paper is organized as follows: in Section 1 we present the concept of finite tree and the logics we are interested in together with their standard interpretation. Section 2 merely states our three axiomatizations. In Section 3, we introduce non standard semantics called *Henkin semantics*, for which our axiomatizations are easily seen to be complete. Section 4 introduces operations on Henkin structures: substructure formation and a general operation of Henkin structures combination. We obtain Feferman-Vaught theorems for this operation by means of Ehrenfeucht-Fraïssé games. In Section 5, we prove *real* completeness (that is, on the restricted class of finite trees). For that purpose, we consider substructures of trees that we call forests and use the general operation discussed in Section 4 to combine a set of forests into one new forest. Our Feferman-Vaught theorems apply to such constructions and we use them in our main proof of completeness, showing that no formula of our language can distinguish Henkin models of our axioms from real finite trees. We also point out that every standard model of our axioms actually *is* a finite tree.

1 Preliminaries

1.1 Finite Trees

A tree is a partially ordered set such that the set of predecessors of any element (or *node*) is well-ordered (a set is well-ordered if all its non-empty subsets have a least element) and there is a unique smallest element called the root. We are interested in *finite node-labeled sibling-ordered trees*: finite trees in which the children of each node are linearly ordered. Also, the nodes can be labeled by unary predicates. We will call these structures *finite trees* for short.

Definition 1 (Finite tree). *Assume a fixed finite set of unary predicate symbols $\{P_1, \ldots, P_n\}$. By a finite tree, we mean a finite structure $\mathfrak{M} = (M, <, \prec, P_1, \ldots, P_n)$, where $(M, <)$ is a tree (with $<$ the descendant relation) and $\prec$ linearly orders the children of each node.*

1.2 Three Extensions of First-Order Logic

In this section, we introduce three extensions of FO: MSO, FO(TC^1) and FO(LFP^1). In the remaining of the paper (unless explicitly stated otherwise), we will always be working with a fixed purely relational vocabulary σ (i.e. with no individual constant or function symbols) and hence, with σ-structures. We assume as usual that we have a countably infinite set of first-order variables. In the case of MSO and FO(LFP^1), we also assume that we have a countably infinite set of set variables. The semantics defined in this section we will refer to as *standard semantics* and the associated structures, as *standard structures.*

We first introduce monadic second order logic, MSO, which is the extension of first-order logic in which we can quantify over the subsets of the domain.

Definition 2 (Syntax and semantics of MSO). *Let At be a first-order atomic formula, x a first-order variable and X a set variable, we define the set of MSO formulas in the following way:*

$$\phi := At \mid Xx \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \neg\phi \mid \exists x\ \phi \mid \exists X\ \phi$$

We use $\forall X\phi$ (resp. $\forall x\phi$) as shorthand for $\neg\exists X\neg\phi$ (resp. $\neg\exists x\neg\phi$). We define the quantifier depth of a MSO formula as the maximal number of first-order and second-order nested quantifiers. We interpret MSO formulas in first-order structures. Like for FO formulas, the truth of MSO formulas in $\mathfrak{M}$ is defined modulo a valuation g of variables as objects. But here, we also have set variables, to which g assigns subsets of the domain. We let $g[a/x]$ be the assignment which differs from g only in assigning a to x (similarly for $g[A/X]$). The truth of atomic formulas is defined by the usual FO clauses plus the following:

$$\mathfrak{M}, g \models Xx \text{ iff } g(x) \in g(X) \text{ for } X \text{ a set variable}$$

The truth of compound formulas is defined by induction, with the same clauses as in FO and an additional one:

$$\mathfrak{M}, g \models \exists X\phi \text{ iff there is } A \subseteq M \text{ such that } \mathfrak{M}, g[A/X] \models \phi$$

The second logic we are interested in is monadic transitive closure logic, FO(TC^1), which extends FO by closing it under the transitive closure of binary definable relations.

Definition 3 (Syntax and semantics of FO(TC^1)). *Let u, v, x, y be first-order variables, $\phi(x, y)$ a FO(TC^1) formula (which, besides x and y, possibly contains other free variables), we define the set of FO(TC^1) formulas in the following way:*

$$\phi := At \mid Xx \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \neg\phi \mid \exists x\ \phi \mid [TC_{xy}\phi(x, y)](u, v)$$

We use $\forall x\phi$ as shorthand for $\neg\exists x\neg\phi$. We define the quantifier depth of a FO(TC^1) formula as the maximal number of nested first-order quantifiers and TC operators. We interpret FO(TC^1) formulas in first-order structures. The notion of assignation and the truth of atomic formulas is defined as in FO. The truth of compound formulas is defined by induction, with the same clauses as in FO and an additional one:

$$\begin{gathered}\mathfrak{M}, g \models [TC_{xy}\phi](u, v)\\ \textit{iff}\\ \textit{for all } A \subseteq M, \textit{ if } g(u) \in A\\ \textit{and for all } a, b \in M, \ a \in A \textit{ and } \mathfrak{M}, g[a/x, b/y] \models \phi(x, y) \textit{ implies } b \in A,\\ \textit{then } g(v) \in A.\end{gathered}$$

Proposition 1. *On standard structures, the following semantical clause for the TC operator is equivalent to the one given above:*

$$\begin{gathered}\mathfrak{M}, g \models [TC_{xy}\phi(x, y)](u, v)\\ \textit{iff}\\ \textit{there exist } a_1 \dots a_n \in M \textit{ with } g(u) = a_1 \textit{ and } g(v) = a_n\\ \textit{and } \mathfrak{M}, g \models \phi(a_i, a_{i+1}) \textit{ for all } 0 < i < n\end{gathered}$$

Proof. Indeed, suppose there is a finite sequence of points $a_1 \dots a_n$ such that $g(u) = a_1$, $g(v) = a_n$, and for each $i < n$, $\mathfrak{M}, g[x/a_i; y/a_{i+1}] \models \phi$. Then for any subset A containing a_1 and which is closed under ϕ, we can show by induction on the length of the sequence $a_1 \dots a_n$ that a_n belongs to A. Now, on the other hand, suppose that there is no finite sequence like described above. To show that there is a subset A of the required form, we simply take A to be the set of all points that "can be reached from u by a finite sequence". By assumption, v does not belong to this set and the set is closed under ϕ.

Intuitively this means that for a formula of the form $[TC_{xy}\phi](u, v)$ to hold on a standard structure, there must be a *finite* "ϕ path" between the points that are named by the variables u and v.

Finally we will also be interested in monadic least fixpoint logic (FO(LFP^1)), which extends FO with set variables and an explicit monadic least fixpoint operator. Consider a FO(LFP^1) formula $\phi(X, x)$ and a structure $\mathfrak{M}$ together with a valuation g. This formula induces an operator F_ϕ taking a set $A \subseteq dom(\mathfrak{M})$ to the set $\{a : \mathfrak{M}, g[a/X, A/X] \models \phi\}$. FO(LFP^1) is concerned with *least fixpoints* of such operators. If ϕ is positive in X (a formula is positive in X whenever X only occurs in the scope of an even number of negations), the operator F_ϕ is monotone (i.e. $X \subseteq Y$ implies $F_\phi(X) \subseteq F_\phi(Y)$). Monotone operators always have a least fixpoint $LFP(F) = \bigcap\{X | F(X) \subset X\}$ (defined as the intersection of all their prefixpoints).

Definition 4 (Syntax and semantics of FO(LFP^1)). *Let X be a set variable, x, y FO variables, ψ, ξ FO(LFP^1) formulas and $\phi(x, X)$ a FO(LFP^1) formula*

positive in X (besides x and X, $\phi(x, X)$ possibly contains other free variables), we define the set of $\mathsf{FO(LFP^1)}$ formulas in the following way:

$$\psi := At \mid Xy \mid \psi \wedge \xi \mid \psi \vee \xi \mid \psi \to \xi \mid \neg\psi \mid \exists x\ \psi \mid [LFP_{x,X}\phi(x, X)]y$$

We use $\forall x\psi$ as shorthand for $\neg\exists x\neg\psi$. We define the quantifier depth of a $\mathsf{FO(LFP^1)}$ formula as the maximal number of nested first-order quantifiers and LFP operators. Again, we can interpret $\mathsf{FO(LFP^1)}$ formulas in first-order structures. The notion of assignation and the truth of atomic formulas are defined similarly as in the MSO case. The truth of compound formulas is defined by induction, with the same clauses as in FO and an additional one:

$$\mathfrak{M}, g \models [LFP_{x,X}\phi]y$$
iff
for all $A \subseteq dom(\mathfrak{M})$, if for all $a \in dom(\mathfrak{M})$, $\mathfrak{M}, g[a/x, A/X] \models \phi(x, X)$ implies $a \in A$,
then $g(y) \in A$.

1.3 Expressive Power

There is a recursive procedure, transforming any $\mathsf{FO(LFP^1)}$ formula ϕ into a MSO formula ϕ' such that $\mathfrak{M}, g \models \phi$ iff $\mathfrak{M}, g \models \phi'$. The interesting clause is $([LFP_{x,X}\phi(x, X)]y)' = \forall X(\forall x(\phi(x, X)' \to Xx) \to Xy)$. (The other ones are all of the same type, e.g. $(\phi \wedge \psi)^* = (\phi^* \wedge \psi^*)$.) This procedure can easily be seen adequate by considering the semantical clause for the LFP operator.

Now there is also a recursive procedure transforming any $\mathsf{FO(TC^1)}$ formula ϕ into a $\mathsf{FO(LFP^1)}$ formula ϕ'' such that $\mathfrak{M}, g \models \phi$ iff $\mathfrak{M}, g \models \phi''$. The interesting clause is $([TC_{xy}\phi](u, v))'' = [LFP_{Xy}y = u \vee \exists x((Xx \wedge \phi(x, y)''))]v$. Let us give an argument for this claim. By Proposition 1 it is enough to show that $[LFP_{Xy}y = u \vee \exists x(Xx \wedge \phi(x, y)'')]v$ holds if and only if there is a finite ϕ'' path from u to v. For the right to left direction, suppose there is such a path $a_1 \dots a_n$ with $g(u) = a_1$ and $g(v) = a_n$. Then, for any subset A of the domain, we can show by induction on i that if for all a_i $(1 \leq i \leq n)$, $a_i = u \vee \exists x((Ax \wedge \phi(x, a_i)'')$ implies $a_i \in A$, then $v \in A$, i.e., $[LFP_{Xy}y = u \vee \exists x((Xx \wedge \phi(x, y)''))]v$ holds. Now for the left to right direction, suppose there is no such ϕ'' path. Consider the set A of all points that can be reached from u by a finite ϕ'' path. By assumption, $\neg Av$ and it holds that $\forall y((y = u \vee \exists x(Ax \wedge \phi(x, y)'')) \to Ay)$, i.e., $\neg[LFP_{Xy}y = u \vee \exists x(Xx \wedge \phi(x, y)'')]v$.

It is known that on arbitrary structures $\mathsf{FO(TC^1)} < \mathsf{FO(LFP^1)} < \mathsf{MSO}$ (see (5)) and on trees $\mathsf{FO(TC^1)} <_{trees} \mathsf{FO(LFP^1)} =_{trees} \mathsf{MSO}$ (see (19) and (16)). It is also known that the (not FO definable) class of finite trees is already definable in $\mathsf{FO(TC^1)}$ (see for instance (12)), which is the weakest of the logics studied here. We provide additional detail in Section 5.3.

2 The Axiomatizations

As many arguments in this paper equally hold for MSO, $\mathsf{FO(TC^1)}$ and $\mathsf{FO(LFP^1)}$, we let $\Lambda \in \{\mathsf{MSO}, \mathsf{FO(TC^1)}, \mathsf{FO(LFP^1)}\}$ and use Λ as a symbol for any one of them.

FO1.	Tautologies of sentential calculus
FO2.	$\vdash \forall x\phi \to \phi_t^x$, where t is substitutable for x in ϕ
FO3.	$\vdash \forall x(\phi \to \psi) \to (\forall x\phi \to \forall x\psi)$
FO4.	$\vdash \phi \to \forall x\phi$, where x does not occur free in ϕ
FO5.	$\vdash x = x$
FO6.	$\vdash x = y \to (\phi \to \psi)$, where ϕ is atomic and ψ is obtained from ϕ by replacing x in zero or more (but not necessarily all) places by y.
Modus Ponens	if $\vdash \phi$ and $\vdash \phi \to \psi$, then $\vdash \psi$
FO Generalization	if $\vdash \phi$, then $\vdash \forall x\phi$

Fig. 1. Axioms and rules of FO

COMP.	$\vdash \exists X\forall x(Xx \leftrightarrow \phi)$, where X does not occur free in ϕ
MSO1.	$\vdash \forall X\phi \to \phi[X/T]$, where T (which is either a set variable or a monadic predicate) is substitutable in ϕ for X.
MSO2.	$\vdash \forall X(\phi \to \psi) \to (\forall X\phi \to \forall X\psi)$
MSO3.	$\vdash \phi \to \forall X\phi$, where X does not occur free in ϕ
MSO Generalization	if $\vdash \phi$, then $\vdash \forall X\phi$

Fig. 2. Axiom and inference rule of MSO

FO(TC^1) axiom	$\vdash [TC_{xy}\phi](u,v) \to ((\psi(u) \wedge \forall x\forall y(\psi(x) \wedge \phi(x,y) \to \psi(y))) \to \psi(v))$ where ψ is any FO(TC^1) formula
FO(TC^1) Generalization	if $\vdash \xi \to ((P(u) \wedge \forall x\forall y(P(x) \wedge \phi(x,y) \to P(y))) \to P(v))$, and P does not occur in ξ, then $\vdash \xi \to [TC_{xy}\phi](u,v)$

Fig. 3. Axiom and inference rule of FO(TC^1)

FO(LFP^1) axiom	$\vdash [LFP_{x,X}\phi]y \to (\forall x(\phi(x,\psi) \to \psi(x)) \to \psi(y))$ where ψ is any FO(LFP^1) formula and $\phi(x,\psi)$ is the result of the replacement in $\phi(x,X)$ of each occurrence of X by ψ (renaming variables when needed)
FO(LFP^1) Generalization	if $\vdash \xi \to (\forall x(\phi(x,P) \to P(x)) \to P(y))$, and P positive in ϕ does not occur in ξ, then $\vdash \xi \to [LFP_{X,x}\phi](y)$

Fig. 4. Axiom and inference rule of FO(LFP^1)

T1.	$\forall xyz(x < y \wedge y < z \rightarrow x < z)$	$<$ is transitive
T2.	$\neg\exists x(x < x)$	$<$ is irreflexive
T3.	$\forall xy(x < y \rightarrow \exists z(x <_{imm} z \wedge z \leq y))$	immediate children
T4.	$\exists x\forall y\neg(y < x)$	there is a root
T5.	$\forall xyz(x < z \wedge y < z \rightarrow x \leq y \vee y \leq x)$	linearly ordered ancestors
T6.	$\forall xyz(x \prec y \wedge y \prec z \rightarrow x \prec z)$	$\prec$ is transitive
T7.	$\neg\exists x(x \prec x)$	$\prec$ is irreflexive
T8.	$\forall xy(x \prec y \rightarrow \exists z(x \prec_{imm} z \wedge z \preceq y))$	immediately next sibling
T9.	$\forall x\exists y(y \preceq x \wedge \neg\exists z(z \prec y))$	there is a least sibling
T10.	$\forall xy((x \prec y \vee y \prec x) \leftrightarrow (\exists z(z <_{imm} x \wedge z <_{imm} y) \wedge x \neq y))$	$\prec$ linearly orders *siblings*
T11.	$\forall xy(x = y \vee x < y \vee y < x \vee \exists x'y'(x' < x \wedge y' < y \wedge (x' \prec y' \vee y' \prec x')))$	connectedness
Ind.	$\forall x(\forall y((x < y \vee x \prec y) \rightarrow \phi(y)) \rightarrow \phi(x)) \rightarrow \forall x\phi(x)$	

where
$\phi(x)$ ranges over Λ-formulas in one free variable x
and
$x <_{imm} y$ is shorthand for $x < y \wedge \neg\exists z(z < y \wedge x < z)$,
$x \prec_{imm} y$ is shorthand for $x \prec y \wedge \neg\exists z(x \prec z \wedge z \prec y)$

Fig. 5. Specific axioms on finite trees

The axiomatization of Λ on finite trees consists of three parts: the axioms of first-order logic, the specific axioms of Λ, and the specific axioms on finite trees.

To axiomatize FO, we adopt the infinite set of axioms and the two rules of inference given in Figure 1 (like in (6), except from the fact that we use a generalization rule). To axiomatize MSO, the axioms and rule of Figure 2 are added to the axiomatization of FO. We call the resulting system $\vdash_{\mathsf{MSO}}$. $COMP.$ stands for "comprehension" by analogy with the comprehension axiom of set theory. $MSO1$ plays a similar role as $FO2$, $MSO2$ as $FO3$ and $MSO3$ as $FO4$. To axiomatize $\mathsf{FO}(\mathsf{TC}^1)$, the axiom and rule of Figure 3 are added to the axiomatization of FO. We call the resulting system $\vdash_{\mathsf{FO}(\mathsf{TC}^1)}$. To axiomatize $\mathsf{FO}(\mathsf{LFP}^1)$, the axiom and rule of Figure 4 are added to the axiomatization of FO. We call the resulting system $\vdash_{\mathsf{FO}(\mathsf{LFP}^1)}$. We are interested in axiomatizing Λ on the class of finite trees. For that purpose we restrict the class of considered structures by adding to $\vdash_\Lambda$ the axioms given in Figure 5. We call the resulting system $\vdash_\Lambda^{tree}$. Note that the induction scheme in Figure 5 allows to reason by induction on *properties definable in* Λ only. Also, for technical convenience, we adopt the following convention:

Definition 5. *Let Γ be a set of Λ-formulas and ϕ a Λ-formula. By $\Gamma \vdash_\Lambda \phi$ we will always mean that there are $\psi_1, \ldots, \psi_n \in \Gamma$ such that $\vdash_\Lambda (\psi_1 \wedge \ldots \wedge \psi_n) \rightarrow \phi$.*

Now the main result of this paper is that on standard structures, the Λ theory of finite trees is completely axiomatized by $\vdash_\Lambda^{tree}$. In the remaining sections we will progressively build a proof of it.

3 Henkin Completeness

As it is well known, MSO, FO(TC^1) and FO(LFP^1) are highly undecidable on arbitrary standard structures (by arbitrary, we mean any sort of structure: infinite trees, arbitrary graphs, partial orders...) and hence not recursively enumerable. So in order to show that our axiomatizations $\vdash_{\Lambda}^{tree}$ are complete on finite trees, we resort to a special trick, already used by Kees Doets in his PhD thesis (4). We proceed in two steps. First, we show three Henkin completeness theorems, based on non standard (so called Henkin) semantics for MSO, FO(TC^1) and FO(LFP^1) (on the general topic of Henkin semantics, see (10), the original paper by Henkin and also (15)). Each semantics respectively extends the class of standard structures with non standard (Henkin) MSO, FO(TC^1) and FO(LFP^1)-structures. By the Henkin completeness theorems, our axiomatic systems $\vdash_{\Lambda}^{tree}$ naturally turn out to be complete on the wider class of their Henkin-models. But by compactness, some of these models are infinite. As a second step, we show in Section 5 that *no Λ-sentence can distinguish between standard and non-standard Λ-Henkin-models among models of our axioms.* This entails that our axioms are complete on the class of (standard) finite trees, i.e., each Λ-sentence valid on this class is provable using $\vdash_{\Lambda}^{tree}$. Now let us point out that Kees Doets was interested in the completeness of *first-order logic* on finite trees. Thus, he was relying on the FO completeness theorem and if he was working with non-standard models of the FO theory of finite trees, he was not concerned with non standard Henkin-structures in our sense. Hence, what makes the originality of the method developed in this paper is its use of Henkin semantics. So let us begin with the concept of Henkin-structure. Such structures are particular cases among structures called *frames* and it is convenient to define frames before defining Henkin-structures.

Definition 6 (Frames). *Let σ be a purely relational vocabulary. A σ-**frame** $\mathfrak{M}$ consists of a non-empty universe $dom(\mathfrak{M})$, an interpretation in $dom(\mathfrak{M})$ of the predicates in σ and a set of admissible subsets $\mathbb{A}_{\mathfrak{M}} \subseteq \wp(dom(\mathfrak{M}))$.*

Whenever $\mathbb{A}_{\mathfrak{M}} = \wp(dom(\mathfrak{M}))$, $\mathfrak{M}$ can be identified to a standard structure. Assignments g into $\mathfrak{M}$ are defined as in standard semantics, except that if X is a set variable, then we require that $g(X) \in \mathbb{A}_{\mathfrak{M}}$.

Definition 7 (Interpretation of Λ-formulas in frames). *Λ-formulas are interpreted in frames as in standard structures, except for the three following clauses. The set quantifier clause of MSO becomes:*

$$\mathfrak{M}, g \models \exists X\phi \text{ iff there is } A \in \mathbb{A}_{\mathfrak{M}_\Gamma} \text{ such that } \mathfrak{M}, g[A/X] \models \phi$$

The TC clause of FO(TC^1) becomes:

$$\mathfrak{M}, g \models [TC_{xy}\phi](u, v)$$
$$\text{iff}$$
$$\text{for all } A \in \mathbb{A}_{\mathfrak{M}}, \text{ if } g(u) \in A$$
$$\text{and for all } a, b \in dom(\mathfrak{M}),\ a \in A \text{ and } \mathfrak{M}, g[x/a, b/y] \models \phi \text{ imply } b \in A,$$
$$\text{then } g(v) \in A.$$

And finally the LFP clause of $\mathsf{FO(LFP^1)}$ *becomes:*

$$\mathfrak{M}, g \models [LFP_{x,X}\phi]y$$
$$\textit{iff}$$
$$\textit{for all } A \in \mathbb{A}_{\mathfrak{M}}\textit{, if for all } a \in dom(\mathfrak{M}),\ \mathfrak{M}, g[a/x, A/X] \models \phi(x, X) \textit{ implies } a \in A,$$
$$\textit{then } g(y) \in A.$$

Definition 8 (Λ-Henkin-Structures). *A Λ-**Henkin-structure** is a frame $\mathfrak{M}$ that is closed under Λ-definability, i.e., for each Λ-formula φ and assignment g into $\mathfrak{M}$:*

$$\{a \in M | \mathfrak{M}, g[a/x] \models \varphi\} \in \mathbb{A}_{\mathfrak{M}}$$

Remark 1. Note that any finite Λ-Henkin-structure is a standard structure, as every subset of the domain is parametrically definable in a finite structure. Hence, non standard Henkin structures are always infinite.

Theorem 1. *Λ is completely axiomatized on Λ-Henkin-structures by $\vdash_\Lambda$, i.e., for every set of Λ-formulas Γ and Λ-formula ϕ, ϕ is true in all Λ-Henkin-structures of Γ if and only if $\Gamma \vdash_\Lambda \phi$.*

Compactness follows directly from Definition 5 and Theorem 1, i.e., a possibly infinite set of Λ-sentences has a model if and only if every finite subset of it has a model. It also follows directly from Theorem 1 that $\vdash_\Lambda^{tree}$ is complete on the class of its Λ-Henkin-models. Nevertheless, by compactness the axioms of $\vdash_\Lambda^{tree}$ are also satisfied on infinite trees. We overcome this problem by defining a slightly larger class of Henkin structures, which we will call *definably well-founded Λ-quasi-trees.*[1]

Definition 9. *A* Λ-quasi-tree *is any Λ-Henkin structure $(T, <, \prec, P_1, \ldots, P_n, \mathbb{A}_T)$ (where $\mathbb{A}_T$ is the set of admissible subsets of T) satisfying the axioms and rules of $\vdash_\Lambda$ and the axioms T1–T11 of Figure 5. A Λ-quasi-tree is* definably well founded *if, in addition, it satisfies all instances of the induction scheme Ind of Figure 5.*

Corollary 1. *A Λ-Henkin-structure satisfies the axioms of $\vdash_\Lambda^{tree}$ if and only if it is a definably well-founded Λ-quasi-tree.*

4 Operations on Henkin-Structures

Let $\Lambda \in \{\mathsf{MSO}, \mathsf{FO(TC^1)}, \mathsf{FO(LFP^1)}\}$. As noted in Remark 1, every finite Λ-Henkin structure is also a standard structure. Hence, when working in finite model theory, it is enough to rely on the usual FO constructions to define operations on structures. On the other hand, even though our main completeness

[1] For a nice picture of a *non* definably well-founded quasi-tree see (1).

result concerns finite trees, inside the proof we need to consider infinite (Λ-Henkin) structures and operations on them. In this context, methods for forming new structures out of existing ones have to be redefined carefully. We first propose a notion of substructure of a Λ-Henkin-structure generated by one of its parametrically definable admissible subsets:

Definition 10 (Λ-substructure). *Let $\mathfrak{M} = (dom(\mathfrak{M}), Pred, \mathbb{A}_{\mathfrak{M}})$ be a Λ-Henkin-structure (where Pred is the interpretation of the predicates). We call $\mathfrak{M}_{FO} = (dom(\mathfrak{M}), Pred)$ the FO-structure underlying $\mathfrak{M}$. Given a parametrically definable set $A \in \mathbb{A}_{\mathfrak{M}}$, the Λ-substructure of $\mathfrak{M}$ generated by A is the structure $\mathfrak{M} \restriction A = (\langle A \rangle_{\mathfrak{M}_{FO}}, \mathbb{A}_{\mathfrak{M} \restriction A})$, where $\langle A \rangle_{\mathfrak{M}_{FO}}$ is the FO-substructure of $\mathfrak{M}_{FO}$ generated by A (note that A forms the domain of $\langle A \rangle_{\mathfrak{M}_{FO}}$, as the vocabulary is purely relational) and $\mathbb{A}_{\mathfrak{M} \restriction A} = \{X \cap A | X \in \mathbb{A}_{\mathfrak{M}}\}$.*

Now, in order to show that Λ-substructures are Henkin-structures, we introduce a notion of *relativization* and a corresponding *relativization lemma*. This lemma establishes that for any Λ-Henkin-structure $\mathfrak{M}$ and Λ-substructure $\mathfrak{M} \restriction A$ of $\mathfrak{M}$ (with A a set parametrically definable in $\mathfrak{M}$), if a set is parametrically definable in $\mathfrak{M} \restriction A$ then it is also parametrically definable in $\mathfrak{M}$. This result will be useful again in Section 5.2.

Definition 11 (Relativization mapping). *Given two Λ-formulas ϕ, ψ having no variables in common and given a FO variable x, we define $REL(\phi, \psi, x)$ by induction on the complexity of ϕ and call it the* relativization of ϕ to ψ*:*

- *If ϕ is an atom, $REL(\phi, \psi, x) = \phi$,*
- *If $\phi :\approx \phi_1 \wedge \phi_2$, $REL(\phi, \psi, x) = REL(\phi_1, \psi, x) \wedge REL(\phi_2, \psi, x)$ (similar for $\vee, \rightarrow, \neg$),*
- *If $\phi :\approx \exists y \chi$, $REL(\phi, \psi, x) = \exists y(\psi[y/x] \wedge REL(\chi, \psi, x))$ (where $\psi[y/x]$ is the formula obtained by replacing in ψ every occurrence of x by y),*
- *If $\phi :\approx \exists Y \chi$, $REL(\phi, \psi, x) = \exists Y((Yx \rightarrow \psi) \wedge REL(\chi, \psi, x))$,*
- *If $\phi :\approx [TC_{yz}\chi](u, v)$, $REL(\phi, \psi, x) = [TC_{yz} REL(\chi, \psi, x) \wedge \psi[y/x] \wedge \psi[z/x]](u, v)$,*
- *If $\phi :\approx [LFP_{Xy}\chi]z$, $REL(\phi, \psi, x) :\approx [LFP_{Xy}\chi \wedge \psi[y/x]]z$.*

Lemma 1 (Relativization lemma). *Let $\mathfrak{M}$ be a Λ-Henkin-structure, g a valuation on $\mathfrak{M}$, ϕ, ψ Λ-formulas and $A = \{x \mid \mathfrak{M}, g \models \psi\}$. If $g(y) \in A$ for every variable y occurring free in ϕ and $g(Y) \in \mathfrak{M} \restriction A$ for every set variable Y occurring free in ϕ, then $\mathfrak{M}, g \models REL(\phi, \psi, x) \Leftrightarrow \mathfrak{M} \restriction A, g \models \phi$.*

Lemma 2. *$\mathfrak{M} \restriction A$ is a Λ-*Henkin*-structure.*

Proof. Take B parametrically definable in $\mathfrak{M} \restriction A$, i.e., there is a Λ-formula $\phi(y)$ and an assignment g such that $B = \{a \in dom(\mathfrak{M} \restriction A) \mid \mathfrak{M} \restriction A, g[a/y] \models \phi(y)\}$. Now we know that A is also parametrically definable in $\mathfrak{M}$, i.e., there is a Λ-formula $\psi(x)$ and an assignment g' such that $A = \{a \in dom(\mathfrak{M}) \mid \mathfrak{M}, g'[a/x] \models \psi(x)\}$. Assume w.l.o.g. that ϕ and ψ have no variables in common, we define an assignment g^* by letting $g^*(z) = g'(z)$ for every variable z occurring in ψ

and $g^*(z) = g(z)$ otherwise. The situation with set variables is symmetric. Now by Lemma 1, $B = \{a \in dom(\mathfrak{M}) \mid \mathfrak{M}, g^*[a/x] \models REL(\phi, \psi, x)\}$ and hence $B \in \mathbb{A}_{\mathfrak{M}\restriction A}$.

There is in model theory a whole range of methods to form new structures out of existing ones. A standard reference on the matter is (7), written in a very general algebraic setting. Familiar constructions like disjoint unions of FO-structures are redefined as particular cases of a new notion of *generalized product* of FO-structures and abstract properties of such products are studied. In particular, an important theorem now called the Feferman-Vaught theorem for FO is proven. We are particularly interested in one of its corollaries, which establishes that generalized products of FO-structures preserve elementary equivalence. This is related to our work in that we show an analogue of this result for a particular case of generalized product of Λ-Henkin-structures that we call *fusion*, this notion being itself a generalization of a notion of disjoint unions of Λ-Henkin-structures that we also define.

Definition 12 (Disjoint union of Λ-Henkin-structures). *Let σ be a purely relational vocabulary and $\sigma^* = \sigma \cup \{Q_1, \ldots, Q_k\}$, with $\{Q_1, \ldots, Q_k\}$ a set of new monadic predicates. For any Λ-Henkin-structures $\mathfrak{M}_1, \ldots, \mathfrak{M}_k$ in vocabulary σ with disjoint domains, define their* disjoint union $\biguplus_{1 \leq i \leq k} \mathfrak{M}_i$ *(or,* direct sum*) to be the σ^*-frame that has as its domain the union of the domains of the structures $\mathfrak{M}_i$ and likewise for the relations, except for the predicates Q_i, whose interpretations are respectively defined as the domain of the structures $\mathfrak{M}_i$ (we will use Q_i to index the elements of M_i). The set of admissible subsets $\mathbb{A}_{\biguplus_{1 \leq i \leq k} \mathfrak{M}_i}$ is the closure under finite union of the union of the sets of admissible subsets of the $\mathfrak{M}_i$. That is:*

- $dom(\biguplus_{1 \leq i \leq k} \mathfrak{M}_i) = \bigcup_{1 \leq i \leq k} dom(\mathfrak{M}_i)$
- $P^{\biguplus_{1 \leq i \leq k} \mathfrak{M}_i} = \bigcup_{1 \leq i \leq k} P^{\mathfrak{M}_i}$ *(with $P \in \sigma$) and* $Q_i^{\biguplus_{1 \leq i \leq k} \mathfrak{M}_i} = dom(\mathfrak{M}_i)$
- $A \in \mathbb{A}_{\biguplus_{1 \leq i \leq k} \mathfrak{M}_i}$ *iff* $A = \bigcup_{1 \leq i \leq k} A_i$ *for some* $A_i \in \mathbb{A}_{\mathfrak{M}_i}$

Definition 13 (f-fusion of Λ-Henkin-structures). *Let σ be a purely relational vocabulary and $\sigma^* = \sigma \cup \{Q_1, \ldots, Q_k\}$, with $\{Q_1, \ldots, Q_k\}$ a set of new monadic predicates. Let f be a function mapping each n-ary predicate $P \in \sigma$ to a quantifier-free formula over σ^* in variables $x_1, \ldots, x_n$. For any Λ-Henkin-structures $\mathfrak{M}_1, \ldots, \mathfrak{M}_k$ in vocabulary σ with disjoint domains, define their* f-fusion *to be the σ-frame $\bigoplus^f_{1 \leq i \leq k} \mathfrak{M}_i$ that has the same domain and set of admissible subsets as $\biguplus_{1 \leq i \leq k} \mathfrak{M}_i$. For any $P \in \sigma$, the interpretation of P in $\bigoplus^f_{1 \leq i \leq k} \mathfrak{M}_i$ is the set of n-tuples satisfying $f(P(x_1 \ldots x_n))$ in $\biguplus_{1 \leq i \leq k} \mathfrak{M}_i$.*

An easy example of f-fusion on standard structures[2] is the ordered sum of two linear orders $(M_1, <_1), (M_2, <_2)$, where all the elements of M_1 are before

[2] It is simpler to give an example on standard structures, because then, we do not have to say anything about admissible sets.

the elements of M_2. In this case, σ consists of a single binary relation $<$, the elements of M_1 are indexed with Q_1, those of M_2 with Q_2 and f maps $<$ to $x < y \vee (Q_1 x \wedge Q_2 y)$.

We show preservation results involving f-fusions of Λ-Henkin-structures. Hence we deal with analogues of elementary equivalence for these logics and we refer to *Λ-equivalence*.

Definition 14. *Given two Λ-Henkin-structures $\mathfrak{M}$ and $\mathfrak{N}$, we write $\mathfrak{M} \equiv_\Lambda \mathfrak{N}$ and say that $\mathfrak{M}$ and $\mathfrak{N}$ are* Λ-equivalent *if they satisfy the same Λ-sentences. Also, for any natural number n, we write $\mathfrak{M} \equiv^n_\Lambda \mathfrak{N}$ and say that $\mathfrak{M}$ and $\mathfrak{N}$ are* n-Λ-equivalent *if $\mathfrak{M}$ and $\mathfrak{N}$ satisfy the same Λ-sentences of quantifier depth at most n. In particular, $\mathfrak{M} \equiv_\Lambda \mathfrak{N}$ holds iff, for all n, $\mathfrak{M} \equiv^n_\Lambda \mathfrak{N}$ holds.*

Now we are ready to introduce our "Feferman-Vaught theorems". Comparable work had already been done by Makowski in (14) for extensions of FO, but a crucial difference is that he only considered standard structures, whereas we need to deal with Λ-Henkin-structures. Our proofs make use of Ehrenfeucht-Fraïssé games for each of the logics Λ. The MSO game, that we show to be adequate, is rather straightforward and has already been used by other authors (see for instance (13)). The FO(LFP^1) game is borrowed from Uwe Bosse (2). It also applies to Henkin structures, as careful inspection of its adequacy proof shows. The FO(TC^1) game has already been mentioned in passing by Grädel in (9) as an alternative to the game he used and we show that it is adequate. It looks also similar to a system of partial isomorphisms given in (3). However it is important to note that this game is different from the FO(TC^1) game which is actually used in (9). The two games are equivalent when played on standard structures, but not when played on FO(TC^1)-Henkin structures. This is so because the game used in (3) relies on the alternative semantics for the TC operator given in Proposition 1, so that only finite sets of points can be chosen by players ; whereas the game we use involves choices of not necessarily finite admissible subsets. These are not equivalent approaches. Indeed, on FO(TC^1)-Henkin structures a simple compactness argument shows that the semantical clause of Proposition 1 (defined in terms of existence of a *finite* path) is not adequate.

Using these games we show that f-fusions of Λ-Henkin-structures preserve Λ-equivalence.

Theorem 2. *Let $\mathfrak{M}_i$ $\mathfrak{N}_i$ with $1 \leq i \leq k$ be Λ-Henkin-structures. For any f such as described in definition 13, whenever $\mathfrak{M}_i \equiv^n_\Lambda \mathfrak{N}_i$ for all $1 \leq i \leq k$, then also $\bigoplus^f_{1 \leq i \leq k} \mathfrak{M}_i \equiv^n_\Lambda \bigoplus^f_{1 \leq i \leq k} \mathfrak{N}_i$.*

Corollary 2. *For any Λ-Henkin-structures $\mathfrak{M}_i$ with $1 \leq i \leq k$, $\bigoplus^f_{1 \leq i \leq k} \mathfrak{M}_i$ is also a Henkin structure.*

Analogues of Theorem 2 and Corollary 2 for disjoint union follow as well.

5 Completeness on Finite Trees

5.1 Forests and Operations on Forests

In Section 5.2, we will prove that no Λ-sentence can distinguish Λ-Henkin-models of $\vdash_\Lambda^{tree}$ from standard models of $\vdash_\Lambda^{tree}$. More precisely, we will show that for each n, any definably well-founded Λ-quasi-tree is n-Λ-equivalent to a finite tree. In order to give an inductive proof, it will be more convenient to consider a stronger version of this result concerning a class of finite and infinite Henkin structures that we call *quasi-forests*. In this section, we give the definition of quasi-forest and we show how they can be combined into bigger quasi-forests using the notion of fusion from Section 4. Whenever quasi forests are finite, we simply call them *finite forests*. As a simple example, consider a finite tree and remove the root node, then it is no longer a finite tree. Instead it is a finite sequence of trees, whose roots stand in a linear (sibling) order.[3] It does not have a unique root, but it does have a unique *left-most root*. For technical reasons it will be convenient in the definition of quasi forests to add an extra monadic predicate R labelling the roots.

Definition 15 (Λ-quasi-forest). *Let $T = (dom(T), <, \prec, P_1, \ldots, \ldots P_n, \mathbb{A}_T)$ be a Λ-quasi-tree. Given a node a in T, consider the Λ-substructure of T generated by the set $\{x \mid \exists z(a \preceq z \wedge z \leq x)\}$, which is the set formed by a together with all its siblings to the right and their descendants. The Λ-quasi-forest T_a is obtained by labeling each root in this substructure with R ($Rx \Leftrightarrow_{def} \neg\exists y\ y < x$). Whenever T is a tree, we simply call T_a a forest.*

We will show in our main proof of completeness that for each n and for each node a in a Λ-Henkin definably well-founded quasi-tree, the Λ-quasi-forest T_a is n-Λ-equivalent to a finite forest. Our proof will use a notion of composition of Λ-quasi-forests which is a special case of fusion. Given a single node forest F_1 and two Λ-quasi-forests F_2 and F_3, we construct a new Λ-quasi-forest $\bigoplus^{COMP}(F_1, F_2, F_3)$ by letting the only element in F_1 be the left-most root, the roots of F_2 become the children of this node and the roots of F_3 become its siblings to the right. We then derive a corollary of Theorem 2 for compositions of Λ-quasi-forests and use it in Section 5.2.

Definition 16. *Let $\sigma = \{<, \prec, R, P_1, \ldots, P_n\}$, be a relational vocabulary with only monadic predicates except $<$ and $\prec$. Given three additional monadic predicates Q_1, Q_2, Q_3, we define a mapping $COMP$ from σ to quantifier-free formulas over $\sigma \cup \{Q_1, Q_2, Q_3\}$ by letting*

- $COMP(x < y) = x < y \vee (Q_1(x) \wedge Q_2(y))$
- $COMP(x \prec y) = x \prec y \vee (Q_1(x) \wedge Q_3(y) \wedge R(y))$
- $COMP(R(x)) = (Q_3(x) \wedge R(x)) \vee Q_1(x))$

Corollary 3. *Let F_1 be a single node forest and F_2, F_3 Λ-quasi forests. If $F_2 \equiv_\Lambda^n F_2'$ and $F_3 \equiv_\Lambda^n F_3'$ then $\bigoplus^{COMP}(F_1, F_2, F_3) \equiv_\Lambda^n \bigoplus^{COMP}(F_1, F_2', F_3')$.*

[3] Note that, as far as roots are concerned, two nodes can be siblings without sharing any parent. This would not happen in a quasi tree.

5.2 Main Proof of Completeness

Lemma 3. *For all $n \in \mathbb{N}$, every definably well-founded Λ-quasi-tree of finite signature is n-Λ-equivalent to a finite tree. In particular, a Λ-sentence is valid on definably well-founded Λ-quasi-trees iff it is valid on finite trees.*

Proof. Let T be a Λ-quasi-tree, w.l.o.g. assume that a monadic predicate R labels its root. During this proof, it will be convenient to work with Λ-quasi-forests. Note that finite Λ-quasi-forests are simply finite forests and finite Λ-quasi-trees are simply finite trees. Let X_n be the set of all nodes a of T for which it holds that T_a is n-Λ-equivalent to a finite forest. We first show that "belonging to X_n" is a property definable in T (Claim 1). Then, we use the induction scheme to show that every node of a definably well-founded Λ-quasi-tree (and in particular, the root) has this property (Claim 2).

Claim 1: X_n is invariant for $n+1$-Λ-equivalence (i.e., $(T,a) \equiv^{\Lambda}_{n+1} (T,b)$ implies that $a \in X_n$ iff $b \in X_n$), and hence is defined by a Λ-formula of quantifier depth $n+1$.

Proof of claim. Suppose that $(T,a) \equiv^{\Lambda}_{n+1} (T,b)$. We will show that $T_a \equiv^{\Lambda}_{n} T_b$, and hence, by the definition of X_n, $a \in X_n$ iff $b \in X_n$. By the definition of Λ-quasi-forests, $dom(T_a) = \{x \mid \exists z(a \preceq z \wedge z \leq x)\}$. Let ϕ be any Λ-sentence of quantifier depth n. We can assume w.l.o.g. that ϕ does not contain the variables z and x (otherwise we can rename in ϕ these two variables). By lemma 1, $(T,a) \models REL(\phi, \exists z(a \preceq z \wedge z \leq x), x)$ iff $T_a \models \phi$. Notice that $REL(\phi, \exists z(a \preceq z \wedge z \leq x), x)$ expresses precisely that ϕ holds in (T,a) within the subforest T_a. Moreover, the quantifier depth of $REL(\phi, \exists z(a \preceq z \wedge z \leq x)$ is at most $n+1$. It follows that $(T,a) \models REL(\phi, \exists z(a \preceq z \wedge z \leq x), x)$ iff $(T,b) \models REL(\phi, \exists z(b \preceq z \wedge z \leq x), x)$, and hence $T_a \models \phi$ iff $T_b \models \phi$.

For the second part of the claim, note that, up to logical equivalence, there are only finitely many Λ-formulas of any given quantifier depth, as the vocabulary is finite. ⊣

Claim 2: If all descendants and siblings to the right of a belong to X_n, then a itself belongs to X_n.

Proof of claim. Let us consider the case where a has both a descendant and a following sibling (all other cases are simpler). Then, by axioms T3, T5, T8, T9 and T10, a has a first child b, and an immediate next sibling c. Moreover, we know that both b and c are in X_n. In other words, T_b and T_c are n-Λ-equivalent to finite forests T'_b and T'_c. Now, we construct a finite Λ-quasi-forest T'_a by taking a $COMP$-fusion of T'_b, T'_c and of the Λ-substructure of T generated by $\{a\}$, which unique element becomes a common parent of all roots of T'_b and a left sibling of all roots of T'_c. So we get $T'_a = \bigoplus^{COMP}(T \restriction \{a\}, T'_b, T'_c))$. It is not hard to see that T'_a is again a finite forest. Moreover, by the fusion lemma, $\bigoplus^{COMP}(T \restriction \{a\}, T_b, T_c)) \equiv^{\Lambda}_{n} T'_a$. Now to show that $\bigoplus^{COMP}(T \restriction \{a\}, T_b, T_c))$ is isomorphic to T_a (which entails $T_a \equiv^{\Lambda}_{n} T'_a$ i.e. T_a is n-Λ-equivalent to a finite forest), it is enough

to show $\mathbb{A}_{T_a} = \mathbb{A}_{\bigoplus^{COMP}(T\restriction\{a\},T_b,T_c)}$. It holds that $\mathbb{A}_{\bigoplus^{COMP}(T\restriction\{a\},T_b,T_c)} \subseteq \mathbb{A}_{T_a}$ because we can define in T_a each such union of sets by means of a disjunction. Now to show $\mathbb{A}_{T_a} \subseteq \mathbb{A}_{\bigoplus^{COMP}(T\restriction\{a\},T_b,T_c)}$, take $A \in \mathbb{A}_{T_a}$, so $A = A_1 \cup A_2 \cup A_3$ with $A_1 \in \mathbb{A}_{T\restriction\{a\}}, A_2 \in \mathbb{A}_{T_b}, A_3 \in \mathbb{A}_{T_c}$. The domain of each of these three structures is definable in T_a, let say ϕ_1 defines $dom(T \restriction \{a\})$, ϕ_2 defines $dom(T_b)$ and ϕ_3 defines $dom(T_c)$. So each A_i component is definable in T_a (just take the conjunction $\phi_i(x) \wedge Ax$). But then A_i was already definable in $\bigoplus^{COMP}(T \restriction \{a\}, T_b, T_c)$ (by construction of this structure). ⊣

It follows from these two claims, by the induction scheme for definable properties, that X_n contains all nodes of the Λ-quasi-tree, including the root, and hence T is n-Λ-equivalent to a finite tree. For the second statement of the lemma, it suffices to note that every Λ-sentence has a finite vocabulary and a finite quantifier depth.

Theorem 3. *Let $\Lambda \in \{\mathsf{MSO}, \mathsf{FO(TC^1)}, \mathsf{FO(LFP^1)}\}$. The Λ-theory of finite trees is completely axiomatized by $\vdash_{\Lambda}^{tree}$.*

Proof. Theorem 3 follows directly from Lemma 3 and Corollary 1.

5.3 The Set of $\vdash_{\Lambda}^{tree}$ Consequences *Defines* the Class of Finite Trees

Proposition 2 shows together with Theorem 3 that on standard structures, the set of $\vdash_{\Lambda}^{tree}$ consequences actually *defines* the class of finite trees. That is, $\vdash_{\Lambda}^{tree}$ has *no infinite standard model* at all.

Proposition 2. *Let $\Lambda \in \{\mathsf{FO(TC^1)}, \mathsf{FO(LFP^1)}, \mathsf{MSO}\}$. On standard structures, there is a Λ-formula which defines the class of finite trees.*

Proof (Sketch of the proof). It is enough to show it for $\Lambda = \mathsf{FO(TC^1)}$. It follows by Section 1.3 that it also holds for MSO and $\mathsf{FO(LFP^1)}$.

We merely give a sketch of the proof. For additional details we refer the reader to (12). It can be shown that on standard structures, the finite conjunction of the axioms T1–T11 in Figure 5 "almost" defines the class of finite trees, i.e. any finite structure satisfying this conjunction is a finite tree. Now we will explain how to construct an other sentence, which together with this one, actually defines on arbitrary standard structures the class of finite trees. Let L be a shorthand for the formula labelling the leaves in the tree ($Lx \Leftrightarrow_{def} \neg\exists y x < y$) and R a shorthand for the formula labelling the root ($Rx \Leftrightarrow_{def} \neg\exists y y < x$). Consider the depth-first left-to-right ordering of nodes in a tree and the $\mathsf{FO(TC^1)}$ formula $\phi(x, y)$ saying "the node that comes after x in this ordering is y":

$$\phi(x,y) :\approx (\neg Lx \wedge x <_{imm} y \wedge \neg\exists z z \prec y) \vee (Lx \wedge x \prec_{imm} y) \vee (Lx \wedge \neg\exists z x \prec z \wedge \exists z(z < x \wedge z \prec_{imm} y \wedge \neg\exists w w < x \wedge z < w \wedge \exists u w \prec_{imm} u))$$

There is also a $\mathsf{FO(TC^1)}$ formula which says that "x is the very last node in this ordering". $\phi(x, y)$ can be combined with this formula into an $\mathsf{FO(TC^1)}$ formula χ expressing that the tree is finite by saying that (we rely here for the interpretation of χ on the alternative semantics for the TC operator given in Proposition 1)

"there is a finite sequence of nodes $x_1 \dots x_n$ such that x_1 is the root, x_{i+1} the node that comes after x_i in the above ordering, for all i, and x_n is the very last node of the tree in the above ordering".

$$\chi :\approx \exists u \exists z (Rz \wedge [TC_{xy}\phi](z,u) \wedge \neg \exists u' (u \neq u' \wedge [TC_{xy}\phi](u,u')))$$

Theorem 4. *The set of* $\vdash_{\Lambda}^{tree}$ *consequences* defines *the class of finite trees.*

Proof. By Proposition 2 we can express in Λ by means of some formula χ that a structure is a finite tree. So χ is necessarily a consequence of $\vdash_{\Lambda}^{tree}$ (as it is a Λ-formula valid on the class of finite trees).

6 Conclusions

In this paper, taking inspiration from Kees Doets (4) we developed a uniform method for obtaining complete axiomatizations of fragments of MSO on finite trees. For that purpose, we had to adapt classical tools and notions from finite model theory to the specificities of Henkin semantics. The presence of admissible subsets called for some refinements in model theoretic constructions such as formation of substructure or disjoint union. Also, we noticed that not every Ehrenfeucht-Fraïssé game that has been used for FO(TC^1) was suitable to use on Henkin-structures. We focused on a game which doesn't seem to have been used previously in the literature. We also elaborated analogues of the FO Feferman-Vaught theorem for MSO, FO(TC^1) and FO(LFP^1). We considered fusions of structures, a particular case of the Feferman-Vaught notion of generalized product and obtained results which might be interesting to generalize and use in other contexts.

We applied our method to MSO, FO(TC^1) and FO(LFP^1), but it would be worth also examining other fragments of MSO, such as monadic deterministic transitive closure logic (FO(DTC^1)) or monadic alternating transitive closure logic (FO(ATC^1)), see also (3).

Finally, an important feature of our main completeness argument is the way we used the inductive scheme of Figure 5. Hence, extending our approach to another class of finite structures would involve finding a comparable scheme. We also know that we should focus on a logic which is decidable on this class, as on finite structures recursive enumerability is equivalent to decidability. This suggests that other natural candidates would be fragments of MSO on classes of finite structures with bounded treewidth.

References

[1] Backofen, R., Rogers, J., Vijay-Shankar, K.: A first-order axiomatization of the theory of finite trees. Journal of Logic, Language and Information 4(4), 5–39 (1995)

[2] Bosse, U.: An Ehrenfeucht-Fraïssé game for fixpoint logic and stratified fixpoint logic. In: Martini, S., Börger, E., Kleine Büning, H., Jäger, G., Richter, M.M. (eds.) CSL 1992. LNCS, vol. 702, pp. 100–114. Springer, Heidelberg (1993)

[3] Calo, A., Makowsky, J.A.: The Ehrenfeucht-Fraïssé games for transitive closure. In: Nerode, A., Taitslin, M.A. (eds.) LFCS 1992. LNCS, vol. 620, pp. 57–68. Springer, Heidelberg (1992)
[4] Doets, K.: Completeness and Definability: Applications of the Ehrenfeucht Game in Second-Order and Intensional Logic. PhD thesis, Universiteit van Amsterdam (1987)
[5] Ebbinghaus, H.-D., Flum, J.: Finite Model Theory. Perspectives in Mathematical Logic. Springer, Berlin (1995)
[6] Enderton, H.: A mathematical introduction to Logic. Academic Press, New York (1972)
[7] Feferman, S., Vaught, R.: The first-order properties of algebraic systems. Fundamenta Mathematicae 47, 57–103 (1959)
[8] Gottlob, G., Koch, C.: Monadic datalog and the expressive power of languages for web information extraction. In: Proceedings of PODS 2002, pp. 17–28 (2002)
[9] Grädel, E.: On transitive closure logic. In: Kleine Büning, H., Jäger, G., Börger, E., Richter, M.M. (eds.) CSL 1991. LNCS, vol. 626, pp. 149–163. Springer, Heidelberg (1992)
[10] Henkin, L.: Completeness in the theory of types. The Journal of Symbolic Logic 15(2), 81–91 (1950)
[11] Kepser, S.: Querying linguistic treebanks with monadic second-order logic in linear time. J. of Logic, Lang. and Inf. 13(4), 457–470 (2004)
[12] Kepser, S.: Properties of binary transitive closure logic over trees. In: Satta, G., Monachesi, P., Penn, G., Wintner, S. (eds.) Formal Grammar 2006, pp. 77–89 (2006)
[13] Lafitte, G., Mazoyer, J.: Théorie des modèles et complexité. Technical report, Ecole Normale Supérieure de Lyon (1998)
[14] Makowsky, J.A.: Algorithmic uses of the Feferman-Vaught theorem. Annals of Pure and Applied Logic 126(1-3), 159–213 (2004)
[15] Manzano, M.: Extensions of First-Order logic. Cambridge University Press, New York (1996)
[16] Matz, O., Schweikardt, N.: Expressive power of monadic logics on words, trees, pictures, and graphs. In: Grädel, E., Flum, J., Wilke, T. (eds.) Logic and Automata: History and Perspectives, Texts in Logic and Games, pp. 531–552. Amsterdam University Press (2007)
[17] Rogers, J.: Descriptive Approach to Language - Theoretic Complexity. CSLI Publications, Stanford (1998)
[18] ten Cate, B., Marx, M.: Axiomatizing the logical core of XPath 2.0. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 134–148. Springer, Heidelberg (2006)
[19] ten Cate, B., Segoufin, L.: XPath, transitive closure logic, and nested tree walking automata. In: PODS 2008: Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp. 251–260. ACM, New York (2008)
[20] Tiede, H.-J., Kepser, S.: Monadic second-order logic and transitive closure logics over trees. Electron. Notes Theor. Comput. Sci. 165, 189–199 (2006)

Tableau-Based Procedure for Deciding Satisfiability in the Full Coalitional Multiagent Epistemic Logic

Valentin Goranko[1] and Dmitry Shkatov[2]

[1] School of Mathematics, University of the Witwatersrand, South Africa
goranko@maths.wits.ac.za
[2] School of Computer Science, University of the Witwatersrand, South Africa
dmitry@cs.wits.ac.za

Abstract. We study the multiagent epistemic logic **CMAEL(CD)** with operators for common and distributed knowledge for all coalitions of agents. We introduce Hintikka structures for this logic and prove that satisfiability in such structures is equivalent to satisfiability in standard models. Using this result, we design an incremental tableau based decision procedure for testing satisfiability in **CMAEL(CD)**.

1 Introduction

Multiagent epistemic logics formalize reasoning about individual and collective knowledge of agents in multiagent systems. Over the last three decades, such logics have emerged as a useful tool for a number of applications in computer science and AI, the most prominent among them being design, specification, and verification of distributed systems ([6], [7], [3], [2], [10]). In this paper, we consider the logic **CMAEL(CD)** (***C**oalitional **M**ulti-**A**gent **E**pistemic **L**ogic with **C**ommon and **D**istributed knowledge*), involving modal operators for *individual* knowledge for each agent[1], as well as operators for *common* and *distributed* knowledge among any (non-empty) *coalition* of agents. Most of the multiagent epistemic logics studied so far only cover fragments of **CMAEL(CD)**; e.g., the logic considered in [3] contains, besides the individual knowledge modalities, the operator of distributed knowledge only for the whole set of agents in the language, while [11] extends that system with common knowledge operator for the whole set of agents. As far as we know, no provably complete deductive system or a decision procedure has been developed so far for **CMAEL(CD)** (see, however, [1], where a tableau-based decision procedure for a BDI logic involving full coalitions of agents is considered).

One of the major issues in applying multiagent epistemic logics to design of distributed systems is the development of algorithms for *constructive satisfiability testing of formulae*, i.e, checking if a given formula is satisfiable and,

[1] The notion of agent used in this paper is an abstract one; for example, agents can be thought of as components of a distributed system.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 197–213, 2009.

if so, constructing a model for it. The main purpose of this paper is to develop a tableau-based algorithm for the constructive satisfiability problem for **CMAEL(CD)**. In the recent precursor [5] to the present paper, we have developed such an algorithm for the multiagent epistemic logic with operators of individual knowledge, as well as common and distributed knowledge for the set of *all agents*. In the present paper, we extend the results of [5] to **CMAEL(CD)**. The main challenge in this extension lies in handling the operators of distributed knowledge parameterized by coalitions of agents. Thus, while the style of the tableau procedure presented herein is similar to the one from [5], the procedure itself (in particular, the termination-ensureing mechanism), as well as the proof of its correctness, are more involved, the latter requiring more sophisticated model-theoretic techniques than those used in [5], building on the ones employed in [3] and [9]. Consequently, the present paper substantially focuses on overcoming the challenges raised by the presence in the language of coalitional distributed knowledge modalities.

The satisfiability-checking algorithms in [5] and the present paper build upon the ideas underlying the incremental tableaux first proposed in [12] and recently adapted to logics of strategic ability in multiagent systems in [4]. Besides our belief that this approach to building decision procedures for logics for multiagent systems is practically most optimal, the uniformity of method and style of the tableaux in [4] and in the present paper is motivated by our intention to eventually integrate them into a tableau-based decision procedure for comprehensive logical systems for reasoning about knowledge, time, and strategic abilities of agents and coalitions in multiagent systems.

The structure of the paper is as follows: Section 2 presents the logic **CMAEL(CD)**; in Sections 3 and 4, we introduce Hintikka structures for the logic and prove that satisfiability in these structures is equivalent to satisfiability in models. In Sections 5 and 6, we present the tableau procedure for **CMAEL(CD)**-satisfiability and sketch proofs of its soundless, completeness, and termination, and briefly estimate its complexity. We conclude with an example illustrating our tableau procedure and brief concluding remarks.

2 Syntax and Semantics of the Logic CMAEL(CD)

The language $\mathcal{L}$ of **CMAEL(CD)** contains a (finite or countable) set $\mathtt{AP}$ of *atomic propositions*, whose arbitrary members we typically denote by $p, q, r, \ldots$; a finite, non-empty set Σ of names of *agents*, whose arbitrary members we typically denote by $a, b \ldots$ and whose subsets, called *coalitions*, we typically denote by $A, B, \ldots$ (possibly with decorations); a sufficient repertoire of the Boolean connectives; and, for every non-empty coalition A, the modal operators $\mathbf{D}_A$ ("*it is distributed knowledge among A that ...*") and $\mathbf{C}_A$ ("*it is common knowledge among A that ...*"). The formulae of $\mathcal{L}$ are thus defined as follows:

$$\varphi := p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \mathbf{D}_A\varphi \mid \mathbf{C}_A\varphi,$$

where p ranges over $\mathtt{AP}$ and A ranges over non-empty subsets of Σ; the set of all such subsets denoted by $\mathcal{P}^+(\Sigma)$. The other Boolean connectives can be defined

as usual. We denote formulae of $\mathcal{L}$ by $\varphi, \psi, \chi, \ldots$ (possibly with decorations) and omit parentheses in formulae whenever it does not result in ambiguity. We write $\varphi \in \mathcal{L}$ to mean that φ is a formula of $\mathcal{L}$.

The distributed knowledge operator $\mathbf{D}_A\varphi$ intuitively means that a "superagent", somebody who knows everything that any of the agents in A knows, can obtain φ as a logical consequence of their knowledge. For example, if agent a knows ψ and agent b knows $\psi \rightarrow \chi$, then $\mathbf{D}_{\{a,b\}}\chi$ is true even though neither a nor b knows χ. The operators of individual knowledge $\mathbf{K}_a\varphi$ ("*agent a knows that φ*"), for $a \in \Sigma$, can then be defined as $\mathbf{D}_{\{a\}}\varphi$, henceforth written $\mathbf{D}_a\varphi$.

The common knowledge operator $\mathbf{C}_A\varphi$ means that φ is "public knowledge" among A, i.e., that every agent in A knows φ, and knows that every agent in A knows φ, etc. For example, it is common knowledge among drivers that green light means "go" and red light means "stop". Formulae of the form $\neg\mathbf{C}_A\varphi$ are referred to as *(epistemic) eventualities*, for the reasons given later on.

Formulae of $\mathcal{L}$ are interpreted over *coalitional multiagent epistemic models*. In this paper, we also need the auxiliary notions of *coalitional multiagent epistemic structures and frames*, which we now define.

Definition 1. *A* coalitional multiagent epistemic structure *(CMAES, for short) is a tuple* $\mathfrak{G} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)})$*, where*

1. *Σ is a finite, non-empty set of agents;*
2. *$S \neq \emptyset$ is a set of* states*;*
3. *for every $A \in \mathcal{P}^+(\Sigma)$, $\mathcal{R}_A^D$ is a binary relation on S;*
4. *for every $A \in \mathcal{P}^+(\Sigma)$, $\mathcal{R}_A^C$ is the reflexive, transitive closure of $\bigcup_{A'\subseteq A} R_{A'}^D$.*

Definition 2. *A* coalitional multiagent epistemic frame *(CMAEF) is a CMAES $\mathfrak{F} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)})$, where each $\mathcal{R}_A^D$ is an equivalence relation satisfying the following condition:*

$$(\dagger)\quad \mathcal{R}_A^D = \bigcap_{a\in A} \mathcal{R}_a^D$$

If condition (†) above is replaced by the following, weaker, one:

$$(\dagger\dagger)\quad \mathcal{R}_A^D \subseteq \mathcal{R}_B^D \text{ whenever } B \subseteq A,$$

then $\mathfrak{F}$ is a coalitional multiagent epistemic pseudo-frame *(pseudo-CMAEF).*

Note that in every (pseudo-)CMAEF $\mathcal{R}_A^D \subseteq \bigcap_{a\in A} \mathcal{R}_a^D$, and hence $\bigcup_{A'\subseteq A} R_{A'}^D = \bigcup_{a\in A} R_a^D$. Thus, condition (4) of Definition 1 is equivalent to requiring that, in (pseudo-)CMAEFs, $\mathcal{R}_A^C$ is the transitive closure of $\bigcup_{a\in A} R_a^D$, for every $A \in \mathcal{P}^+(\Sigma)$. Moreover, each $\mathcal{R}_A^C$ in a (pseudo-)CMAEF is an equivalence relation.

Definition 3. *A* coalitional multiagent epistemic model *(CMAEM) is a tuple $\mathcal{M} = (\mathfrak{F}, \mathtt{AP}, L)$, where $\mathfrak{F}$ is a CMAEF, $\mathtt{AP}$ is a set of atomic propositions, and $L : S \mapsto \mathcal{P}(\mathtt{AP})$ is a* labeling function*, assigning to every state s the set $L(s)$ of atomic propositions true at s. If $\mathfrak{F}$ is a pseudo-CMAEF, then $\mathcal{M} = (\mathfrak{F}, \mathtt{AP}, L)$ is a* multiagent coalitional pseudo-model *(pseudo-CMAEM).*

The satisfaction relation between (pseudo-)CMAEMs, states, and formulae is defined in the standard way. In particular,

- $\mathcal{M}, s \Vdash \mathbf{D}_A\varphi$ iff $(s,t) \in \mathcal{R}_A^D$ implies $\mathcal{M}, t \Vdash \varphi$;
- $\mathcal{M}, s \Vdash \mathbf{C}_A\varphi$ iff $(s,t) \in \mathcal{R}_A^C$ implies $\mathcal{M}, t \Vdash \varphi$.

Definition 4. *Given a (pseudo-)CMAEM $\mathcal{M}$ and $\varphi \in \mathcal{L}$, we say that φ is* satisfiable *in $\mathcal{M}$ if $\mathcal{M}, s \Vdash \varphi$ holds for some $s \in \mathcal{M}$ and say that φ is* valid *in $\mathcal{M}$ if $\mathcal{M}, s \Vdash \varphi$ holds for every $s \in \mathcal{M}$. Satisfiability and validity in a class of (pseudo-)models are defined accordingly.*

The truth condition for the operator $\mathbf{C}_A$ can be re-stated in terms of reachability. Let $\mathfrak{F}$ be a (pseudo-)CMAEF with state space S and let $s, t \in S$. We say that t is *A-reachable from s* if either $s = t$ or, for some $n \geq 1$, there exists a sequence $s = s_0, s_1, \ldots, s_{n-1}, s_n = t$ of elements of S such that, for every $0 \leq i < n$, there exists $a_i \in A$ such that $(s_i, s_{i+1}) \in R_{a_i}^D$. It is then easy to see that the following truth condition for $\mathbf{C}_A$ is equivalent to the one given above:

- $\mathcal{M}, s \Vdash \mathbf{C}_A\varphi$ iff $\mathcal{M}, t \Vdash \varphi$ whenever t is A-reachable from s.

Note also, that if $\Sigma = \{a\}$, then $\mathbf{D}_a\varphi \leftrightarrow \mathbf{C}_a\varphi$ is valid for all $\varphi \in \mathcal{L}$. Thus, the single-agent case is trivialized and, therefore, we assume throughout the remainder of the paper that Σ contains at least 2 agents.

3 Hintikka Structures

Despite our ultimate interest in satisfiability of (finite sets of) formulae in CMAEMs, the tableaux we present check for the existence of a more general kind of semantic structure for the input set of formulae than a model, namely a *Hintikka structure*. In this section, we introduce Hintikka structures and show that these structures satisfy the same sets of formulae as pseudo-CMAEMs; in the next section, we show that CMAEMs satisfy the same sets of formulae as pseudo-CMAEMs. Consequently, testing for satisfiability in a Hintikka structure can replace testing for satisfiability in a CMAEM. In the following discussion, for brevity, we only consider single formulae; the extension to finite sets of formulae is straightforward.

The most fundamental difference between (pseudo-)models and Hintikka structures is that while the former specify the truth value of every formula of $\mathcal{L}$ at each state, the latter only do so for the formulae relevant to the evaluation of a fixed formula θ. Another important difference is that the D-accessibility relations in (pseudo-) models must satisfy the explicitly stated conditions of Definition 2, while in Hintikka structures conditions are only imposed on the labels of the states in such a way that every Hintikka structure generates, through the construction of Lemma 2 below, a pseudo-CMAEM so that the "truth" of formulae in the labels is preserved in the resultant pseudo-model. To define Hintikka structures, we need the following auxiliary notion.

Definition 5. *A set $\Delta \subseteq \mathcal{L}$ is* fully expanded *if it satisfies the following conditions ($\mathsf{Sub}(\psi)$ stands for the set of subformulas of ψ):*

1. *if* $\neg\neg\varphi \in \Delta$, *then* $\varphi \in \Delta$;
2. *if* $\varphi \wedge \psi \in \Delta$, *then* $\varphi \in \Delta$ *and* $\psi \in \Delta$;
3. *if* $\neg(\varphi \wedge \psi) \in \Delta$, *then* $\neg\varphi \in \Delta$ *or* $\neg\varphi \in \Delta$;
4. *if* $\mathbf{D}_A\varphi \in \Delta$, *then* $\mathbf{D}_{A'}\varphi \in \Delta$ *for every* A' *such that* $A \subseteq A' \subseteq \Sigma$;
5. *if* $\mathbf{D}_A\varphi \in \Delta$, *then* $\varphi \in \Delta$;
6. *if* $\mathbf{C}_A\varphi \in \Delta$, *then* $\mathbf{D}_a(\varphi \wedge \mathbf{C}_A\varphi) \in \Delta$ *for every* $a \in A$;
7. *if* $\neg\mathbf{C}_A\varphi \in \Delta$, *then* $\neg\mathbf{D}_a(\varphi \wedge \mathbf{C}_A\varphi) \in \Delta$ *for some* $a \in A$;
8. *if* $\psi \in \Delta$ *and* $\mathbf{D}_A \in \mathsf{Sub}(\psi)$, *then either* $\mathbf{D}_A \in \Delta$ *or* $\neg\mathbf{D}_A \in \Delta$.

If the condition 4 is dropped, then Δ *is simply* expanded.

Definition 6. *A* coalitional multi-agent epistemic Hintikka structure *(CMAEHS for short) is a tuple* $(\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, H)$ *such that* $(\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)})$ *is a CMAES, and* H *is a labeling of the elements of* S *with sets of formulae of* $\mathcal{L}$ *satisfying the following constraints:*

H1 *if* $\neg\varphi \in H(s)$, *then* $\varphi \notin H(s)$, *for every* $s \in S$;
H2 $H(s)$ *is fully expanded, for every* $s \in S$;
H3 *if* $\neg\mathbf{D}_A\varphi \in H(s)$, *then* $(s,t) \in \mathcal{R}_A^D$ *and* $\neg\varphi \in H(t)$ *for some* $t \in S$;
H4 *if* $(s,t) \in \mathcal{R}_A^D$, *then* $\mathbf{D}_{A'}\varphi \in H(s)$ *iff* $\mathbf{D}_{A'}\varphi \in H(t)$, *for every* $A' \subseteq A$;
H5 *if* $\neg\mathbf{C}_A\varphi \in H(s)$, *then* $(s,t) \in \mathcal{R}_A^C$ *and* $\neg\varphi \in H(t)$ *for some* $t \in S$.

Definition 7. *Let* $\theta \in \mathcal{L}$, $\Theta \subseteq \mathcal{L}$, *and* $\mathcal{H}$ *be a CMAEHS with state space* S. *We say that* $\mathcal{H}$ *is a* CMAEHS for θ, *or that* θ *is* satisfiable *in* $\mathcal{H}$, *if* $\theta \in H(s)$ *for some* $s \in S$; *we say that* Θ *is satisfiable in* $\mathcal{H}$ *if* $\Theta \subseteq H(s)$.

We now prove that $\theta \in \mathcal{L}$ is satisfiable in a pseudo-CMAEM iff there exists a CMAEHS for θ. Given a pseudo-CMAEM $\mathcal{M} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L)$, we define the *extended labeling function* $L^+ : S \mapsto \mathcal{P}(\mathcal{L})$ on $\mathcal{M}$ as follows: $L^+(s) = \{\, \varphi \mid \mathcal{M}, s \Vdash \varphi \,\}$. It is routine to check the following.

Lemma 1. *Let* $\mathcal{M} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L)$ *be a pseudo-CMAEM satisfying* θ *and let* L^+ *be the extended labeling on* $\mathcal{M}$. *Then,* $(\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, L^+)$ *is a CMAEHS for* θ.

Now, we argue in the opposite direction.

Lemma 2. *Let* $\theta \in \mathcal{L}$ *be such that there exists a CMAEHS for* θ. *Then,* θ *is satisfiable in a pseudo-CMAEM.*

Proof. Let $\mathcal{H} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, H)$ be an CMAEHS for θ. We construct a pseudo-CMAEM $\mathcal{M}'$ satisfying θ out of $\mathcal{H}$ as follows.

First, for every $A \in \mathcal{P}^+(\Sigma)$, let $\mathcal{R}_A'^D$ be the reflexive, symmetric, and transitive closure of $\bigcup_{A\subseteq B}\mathcal{R}_B^D$ and let $\mathcal{R}_A'^C$ be the transitive closure of $\bigcup_{a\in A}\mathcal{R}_a'^D$. Notice that $\mathcal{R}_A^D \subseteq \mathcal{R}_A'^D$ and $\mathcal{R}_A^C \subseteq \mathcal{R}_A'^C$ for every $A \in \mathcal{P}^+(\Sigma)$. Second, let $L(s) = H(s) \cap \mathtt{AP}$, for every $s \in S$. It is then easy to check that $\mathcal{M}' = (\Sigma, S, \{\mathcal{R}_A'^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A'^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L)$ is a pseudo-CMAEM.

To complete the proof of the lemma, we show, by induction on the structure of $\chi \in \mathcal{L}$ that, for every $s \in S$ and every $\chi \in \mathcal{L}$, the following hold:

(i) $\chi \in H(s)$ implies $\mathcal{M}', s \Vdash \chi$;
(ii) $\neg\chi \in H(s)$ implies $\mathcal{M}', s \Vdash \neg\chi$.

Let χ be some $p \in \mathtt{AP}$. Then, $p \in H(s)$ implies $p \in L(s)$ and, thus, $\mathcal{M}', s \Vdash p$; if, on the other hand, $\neg p \in H(s)$, then due to (H1), $p \notin H(s)$ and thus $p \notin L(s)$; hence, $\mathcal{M}', s \Vdash \neg p$.

Assume that the claim holds for all subformulae of χ; then, we have to prove that it holds for χ, as well.

Suppose that χ is $\neg\varphi$. If $\neg\varphi \in H(s)$, then the inductive hypothesis immediately gives us $\mathcal{M}', s \Vdash \neg\varphi$; if, on the other hand, $\neg\neg\varphi \in H(s)$, then by virtue of (H2), $\varphi \in H(s)$ and hence, by inductive hypothesis, $\mathcal{M}', s \Vdash \varphi$ and thus $\mathcal{M}', s \Vdash \neg\neg\varphi$.

The case of $\chi = \varphi \wedge \psi$ is straightforward, using (H2).

Suppose that χ is $\mathbf{D}_A\varphi$. Assume, first, that $\mathbf{D}_A\varphi \in H(s)$. In view of the inductive hypothesis, it suffices to show that $(s,t) \in \mathcal{R}'^D_A$ implies $t \in H(t)$. So, assume that $(s,t) \in \mathcal{R}'^D_A$. There are two cases to consider. If $s = t$, then the conclusion immediately follows from (H2). If, on the other hand, $s \neq t$, then there exists an undirected path from s to t along the relations $\mathcal{R}^D_{A'}$, where each A' is a superset of A. Then, in view of (H4), $\mathbf{D}_A\varphi \in H(t)$; hence, by (H2), $\varphi \in H(t)$, as desired.

Assume, next, that $\neg\mathbf{D}_A\varphi \in H(s)$. In view of the inductive hypothesis, it suffices to show that there exist $t \in S$ such that $(s,t) \in \mathcal{R}'^D_A$ and $\neg\varphi \in H(t)$. By (H3), there exists $t \in S$ such that $(s,t) \in \mathcal{R}^D_A$ and $\neg\varphi \in H(t)$. As $\mathcal{R}^D_A \subseteq \mathcal{R}'^D_A$, the desired conclusion follows.

Suppose now that χ is $\mathbf{C}_A\varphi$. Assume that $\mathbf{C}_A\varphi \in H(s)$. In view of the inductive hypothesis, it suffices to show that if t is A-reachable from s in $\mathcal{M}'$, then $\varphi \in H(t)$. So, assume that either $s = t$ or, for some $n \geq 1$, there exists a sequence of states $s = s_0, s_1, \ldots, s_{n-1}, s_n = t$ such that, for every $0 \leq i < n$, there exists $a_i \in \Sigma$ such that $(s_i, s_{i+1}) \in \mathcal{R}'^D_{a_i}$. In the former case, the desired conclusion follows from (H2). In the latter case, we can show by induction on $0 \leq i < n$ that $\mathbf{D}_{a_i}(\varphi \wedge \mathbf{C}_A\varphi) \in H(s_i)$. Then, $\mathbf{D}_{a_{n-1}}(\varphi \wedge \mathbf{C}_A\varphi) \in H(s_{n-1})$, and thus, in view of (H4) and (H2), $\varphi \in H(t)$.

Assume, on the other hand, that $\neg\mathbf{C}_A\varphi \notin H(s)$. Then, the desired conclusion follows from (H6), the fact that $\mathcal{R}^C_A \subseteq \mathcal{R}'^C_A$, and the inductive hypothesis. □

Theorem 1. *Let $\theta \in \mathcal{L}$. Then, θ is satisfiable in a pseudo-CMAEM iff there exists a CMAEHS for θ.*

Proof. Immediately follows from Lemmas 1 and 2. □

4 Equivalence of CMAEMs and Pseudo-CMAEMs

In the present section, we prove that pseudo-CMAEMs and CMAEMs satisfy the same sets of formulae. The right-to-left direction is immediate, as every CMAEM is a pseudo-CMAEM. For the left-to-right direction, we use a modification of the construction from [3, appendix A1] to show that if $\theta \in \mathcal{L}$ is satisfiable in a pseudo-CMAEM, then it is satisfiable in a "tree-like" pseudo-CMAEM that actually turns out to be a bona-fide CMAEM.

Definition 8. *Let* $\mathcal{M} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L)$ *be a (pseudo-) CMAEM and let* $s, t \in S$. *A* maximal path from s to t *in* $\mathcal{M}$ *is a sequence* $s = s_0, A_0, s_1, \ldots, s_{n-1}, A_{n-1}, s_n = t$ *such that, for every* $0 \leq i < n$, $(s_i, s_{i+1}) \in \mathcal{R}_{A_i}^D$, *but* $(s_i, s_{i+1}) \in \mathcal{R}_B^D$ *does not hold for any* B *with* $A_i \subset B \subseteq \Sigma$. *A segment* ρ' *of a maximal path* ρ *starting and ending with a state is a* sub-path *of* ρ.

Definition 9. *Let* $\rho = s_0, A_0 \ldots, A_{n-1}, s_n$ *be a maximal path in* $\mathcal{M}$. *The* reduction *of* ρ *is obtained by, first, replacing in* ρ *every longest sub-path* $s_p, A_p, s_{p+1}, \ldots, A_{p+q-1}, s_{p+q}$ *such that* $s_p = s_{p+1} = \ldots = s_{p+q}$ *with* s_p *(i.e., removing loops) and, then, by replacing in the resultant path every longest sub-path* $s_j, A_j, s_{j+1}, \ldots, A_{j+m-1}, s_{j+m}$ *such that* $A_j = A_{j+1} = \ldots = A_{j+m-1}$ *with* s_j, A_j, s_{j+m} *(i.e., collapsing multiple consecutive transitions along the same relation with a single transition). A maximal path is* reduced *if it equals its own reduction.*

Definition 10. *A (pseudo-)CMAEM* $\mathcal{M}$ *is* tree-like *if, for every* $s, t \in \mathcal{M}$, *there exists at most one reduced maximal path from* s *to* t.

Lemma 3. *If* $\theta \in \mathcal{L}$ *is satisfiable in a pseudo-CMAEM, then it is satisfiable in a (tree-like) CMAEM.*

Proof. Suppose that θ is satisfied in a pseudo-CMAEM $\mathcal{M} = (\Sigma, S, \{\mathcal{R}_A^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L)$ at state s. To build a tree-like CMAEM satisfying θ, we use a modification of the tree-unraveling. The only difference between our construction and the standard tree-unraveling is that the state space of our tree model is made up of all *maximal* paths in $\mathcal{M}$ rather than all paths whatsoever.

Let $\mathcal{M}' = (\Sigma, S', \{\mathcal{R}_A'^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A'^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L')$ be the submodel of $\mathcal{M}$ generated by s. Then, $\mathcal{M}', s \Vdash \theta$ since $\mathcal{M}$ and $\mathcal{M}'$ are locally bisimilar at s. Next, we unravel $\mathcal{M}'$ into a model $\mathcal{M}'' = (\Sigma, S'', \{\mathcal{R}_A''^D\}_{A\in\mathcal{P}^+(\Sigma)}, \{\mathcal{R}_A''^C\}_{A\in\mathcal{P}^+(\Sigma)}, \mathtt{AP}, L'')$ as follows. First, call a maximal path ρ in $\mathcal{M}'$ an s-max-path if the first component of ρ is s. Denote the last element of ρ by $l(\rho)$. Notice that s by itself is an s-max-path. Now, let S'' be the set of all (not necessarily reduced) s-max-paths in $\mathcal{M}'$. For every $A \in \mathcal{P}^+(\Sigma)$, let $\mathcal{R}^{*D}_A = \{\, (\rho, \rho') \mid \rho, \rho' \in S'' \text{and } \rho' = \rho, A, l(\rho') \,\}$ and let, furthermore, $\mathcal{R}_A''^D$ to be the reflexive, symmetric, and transitive closure of $\mathcal{R}^{*D}_A$. Notice that $(\rho, \rho') \in \mathcal{R}_A''^D$ holds iff one of the paths ρ and ρ' extends the other by a sequence of A-steps. Therefore, two different states in S'' can only be connected by $\mathcal{R}_A''^D$ for at most one maximal coalition A. Further, we stipulate the following *downwards closure condition*: whenever $(\rho, \tau) \in \mathcal{R}_A''^D$ and $B \subseteq A$, then $(\rho, \tau) \in \mathcal{R}_B''^D$. The relations $\mathcal{R}_A''^C$ are then defined as in any CMAEF. To complete the definition of $\mathcal{M}''$, we put $L''(\rho) = L'(l(\rho))$, for every $\rho \in S''$.

It is clear from the construction (namely, from the above downwards closure condition) that $\mathcal{M}''$ is a pseudo-CMAEM. We now show that it actually is a (tree-like) CMAEM and that it satisfies θ. To prove the first part of the claim, we need some extra terminology.

We call a maximal path $\rho_1, A_1, \rho_2, \ldots, A_{n-1}, \rho_n$ in $\mathcal{M}''$ *primitive* if, for every $0 \leq i < n$, either $(\rho_i, \rho_{i+1}) \in \mathcal{R}^{*D}_{A_i}$ or $(\rho_{i+1}, \rho_i) \in \mathcal{R}^{*D}_{A_i}$. A primitive path $\rho_1, A_1, \rho_2, \ldots, A_{n-1}, \rho_n$ is *non-redundant* if there is no $0 \leq i < n$ such that

$\rho_i = \rho_{i+2}$ and $A_i = A_{i+1}$. Intuitively, in a non-redundant path we never go from a state ρ (forward or backward) along a relation and then immediately back to ρ along the same relation. Since the relations $\mathcal{R}^{*D}_A$ are edges of a tree, it immediately follows that:

(‡) for every pair of states $\rho, \tau \in S''$, there exists at most one non-redundant primitive path from ρ to τ.

Lastly, we call a primitive path $\rho_1, A, \rho_2, \ldots, A, \rho_n$ an *A-primitive path.*

We will now show that maximal reduced paths in $\mathcal{M}''$ stand in one-to-one correspondence with non-redundant primitive paths. It will then follow from (‡) that maximal reduced paths between any two states of $\mathcal{M}''$ are unique, and thus $\mathcal{M}''$ is tree-like, as claimed. Let $P = \rho_1, A_1, \ldots, A_{n-1}, \rho_n$, where $\rho_1 = \rho$ and $\rho_n = \tau$, be a maximal reduced path from ρ to τ in $\mathcal{M}''$. Since $(\rho_i, \rho_{i+1}) \in \mathcal{R}''^{D}_{A_i}$, there exists a non-redundant A_i-primitive path from ρ_i to ρ_{i+1}, which in view of (‡) is unique. Let us obtain a path P' from ρ to τ by replacing in P every link $(\rho_i, A_i, \rho_{i+1})$ by the corresponding non-redundant A_i-primitive path from ρ_i to ρ_{i+1}. Call P' an *expansion* of P. In view of (‡), every path has a unique expansion. Now, it is easy to see that P is a reduction of P'. Since the reduction of a given path is unique, too, it follows that there exists a one-to-one correspondence between maximal reduced paths and non-redundant primitive paths in $\mathcal{M}''$.

Next, we prove that $\mathcal{R}''^{D}_A = \bigcap_{a \in A} \mathcal{R}''^{D}_a$ for every $A \in \mathcal{P}^+(\Sigma)$, and hence $\mathcal{M}''$ is a CMAEM. The left to right inclusion is immediate from the construction (namely, the downward saturation condition). For the opposite direction, assume that $(s,t) \in \mathcal{R}''^{D}_a$ holds for every $a \in A$. Then, for every $a \in A$, there exists a path, and therefore a maximal reduced path, from s to t along relations $\mathcal{R}^{*D}_{A'}$ such that $a \in A'$. As $\mathcal{M}''$ is tree-like, there is only one maximal reduced path from s to t. Therefore, the relations $\mathcal{R}^{D}_{A'}$ linking s to t along this path are such that $A \subseteq A'$ for every A'. Then, by the downwards closure condition, there is a path from s to t along the relation $\mathcal{R}''^{D}_A$ and, hence, $(s,t) \in \mathcal{R}''^{D}_A$, as desired.

Finally, it remains to prove that $\mathcal{M}''$ satisfies θ. First, notice that $(\rho, \tau) \in \mathcal{R}''_A$ iff there exists an A-primitive path from ρ to τ; hence, as every $\mathcal{R}'_A$ is an equivalence relation, if $(\rho, \tau) \in \mathcal{R}''_A$, then $(l(\rho), l(\tau)) \in \mathcal{R}'_A$. It is now easy to check that the relation $Z = \{ (\rho, l(\rho) \mid \rho \in S'' \}$ is a bisimulation between $\mathcal{M}''$ and $\mathcal{M}'$. Since $(s, l(s)) \in Z$, it follows that $\mathcal{M}'', s \Vdash \theta$, and we are done. □

Theorem 2. *Let $\theta \in \mathcal{L}$. Then, θ is satisfiable in a CMAEM iff there exists a Hintikka structure for θ.*

Proof. Immediate from Theorem 1 and Lemma 3. □

5 Tableaux for CMAEL(CD)

5.1 Basic Ideas and Overview of the Tableau Procedure

The tableau procedure for testing a formula $\theta \in \mathcal{L}$ for satisfiability attempts to construct a non-empty graph $\mathcal{T}^\theta$ (called *tableau*) representing all possible

CMAEHSs for θ. The procedure consists of three major phases: *construction*, *prestate elimination*, and *state elimination*. During the construction phase, we build the *pretableau* $\mathcal{P}^\theta$—a directed graph whose nodes are sets of formulae of two types: *states* and *prestates*[2]. States represent (labels of) states of the CMAEHSs that the tableau attempts to construct, while prestates are "embryo states", expanded into states in the course of the construction. Formally, states are expanded (recall Definition 5), while prestates need not be so. During the construction phase, we produce a directed graph $\mathcal{P}^\theta$—called the *pretableau* for θ—whose set of nodes properly contains the set of nodes of the tableau $\mathcal{T}^\theta$ we are building. During the prestate elimination phase, we create a smaller graph $\mathcal{T}_0^\theta$ out of $\mathcal{P}^\theta$, called the *initial tableau for* θ, by eliminating all the prestates of $\mathcal{P}^\theta$ and adjusting its edges, as prestates have already fulfilled their role and can be discharged. Finally, during the state elimination phase, we remove from $\mathcal{T}_0^\theta$ all states, if any, that cannot be satisfied in any CMAEHS, for one of the three reasons discussed below. The elimination procedure results in a (possibly empty) subgraph $\mathcal{T}^\theta$ of $\mathcal{T}_0^\theta$, called the *final tableau for* θ. If some state Δ of $\mathcal{T}^\theta$ contains θ, we declare θ satisfiable; otherwise, we declare it unsatisfiable.

The philosophy underlying our tableau algorithm stems from the one underpinning the tableau procedure for **LTL** from [12], also recently adapted to Alternating-time temporal logic **ATL** in [4] and to a fragment of **CMAEL(CD)** in [5]. The constructions from these papers, however, can not be directly transferred to the case of **CMAEL(CD)**, for reasons explained below.

Usually, a tableau checks for satisfiability by decomposing the input formula into "simpler" ones. In the classical propositional case, "simpler" implies shorter, thus ensuring the termination of the procedure. Another feature of the tableaux for classical propositional logic is that the decomposition into simpler formulae results in a tree representing an exhaustive search for a Hintikka set (the classical analogue of Hintikka structures) for the input formula θ. If at least one leaf of the tree turns out to be a Hintikka set for θ, it is pronounced satisfiable.

These two features of the classical tableau method do not apply to logics containing fixed point operators, such as $\mathbf{C}_A$.

First, decomposing of a formula $\mathbf{C}_A\varphi$ produces formulae of the form $\mathbf{D}_a(\varphi \wedge \mathbf{C}_A\varphi)$, which are not simpler than the initial formula; rather, they represent the unfolding of the monotone operator whose fixed point is $\mathbf{C}_A\varphi$. Hence, we cannot take termination for granted and need to put a mechanism in place that would guarantee it. In our tableaux, this mechanism is based on the judicious reuse of (pre)states. Unlike the tableaux from [12] and [4], where these are reused without any restrictions, reusing (pre)states in tableaux for **CMAEL(CD)** is not always safe; hence, we will have to place some conditions on such reuse. These conditions, while strict enough to make reusing (pre)states safe, are permissive enough to guarantee the termination of the procedure.

[2] We use "(pre)state" to refer to either state or prestate. The term "(pre)state" denotes both a node of the tableau, which is a set of formulae, and that same set of formulae. Since set-theoretically identical sets may appear as different nodes in the tableau, to avoid ambiguity, we sometimes refer to the sets as "labels" of given (pre)states.

Second, in the classical case, the only reason why the tableau might fail to produce a Hintikka set for the input formula is that every attempt to build such a set results in a collection of formulae containing a *patent inconsistency*—a pair of formulae $\varphi, \neg\varphi$. In the case of **CMAEL(CD)**, there are two other reasons, as the tableau is meant to represent CMAEHSs, more complicated structures than Hintikka sets. One reason has to do with eventualities: the presence of an eventuality $\neg\mathbf{C}_A\varphi$ in the label of a state s of a CMAEHS $\mathcal{H}$ requires the existence in $\mathcal{H}$ of an A-path from s to a state t whose label contains $\neg\varphi$. The analogue of this condition in the tableau is called *realization of eventualities*. Thus, all eventualities in a tableau should be realized in order for the tableau to be "good", i.e., to eventually produce a Hintikka structure. The other reason for a tableau to be "bad" has to do with successor nodes—it may so happen that some of the successors of a node s, necessary for the satisfaction of s, turn out to be unsatisfiable; a "good" tableau can not contain such a node.

5.2 Construction Phase

As already mentioned, the tableau procedure attempts to produce a compact representation of all possible CMAEHSs for the input formula; in this attempt, it organizes an exhaustive search for such CMAEHSs. The pretableau $\mathcal{P}^\theta$ built at this phase contains two types of edge, as well as two types of node (states and prestates).

One type of an edge, depicted by unmarked double arrows $\Longrightarrow$, represents the expansion of the tableau as a search tree. The exhaustive search considers all possible alternatives arising when prestates are expanded into states by branching in the disjunctive cases. Thus, when we draw a double arrow from a prestate Γ to states Δ and Δ' (depicted as $\Gamma \Longrightarrow \Delta$ and $\Gamma \Longrightarrow \Delta'$, respectively), this intuitively means that, in any CMAEHS, a state satisfying Γ has to satisfy at least one of Δ and Δ'.

The second type of an edge represents transition relations in the CMAEHSs which the procedure attempts to build. Accordingly, this type of edge is represented by single, bi-directed (as such transitions are ultimately meant to give rise to symmetric relations) arrows marked with formulae whose presence in one of the end nodes requires the presence in the tableau of the other end node, reachable by a particular relation. All such formulae have the form $\neg\mathbf{D}_A\varphi$ (as can be seen from Definition 6). Intuitively, if $\neg\mathbf{D}_A\varphi \in \Delta$, then some prestate Γ containing $\neg\varphi$ must be accessible from Δ by $\mathcal{R}^D_A$; the reason we mark this single arrow not just by the coalition A, but by formula $\neg\mathbf{D}_A\varphi$, is that it helps us remember not just what relation connects states satisfying Δ and Γ, but why we had to create this particular Γ. This information will be needed when we start eliminating prestates and then states.

Definition 11. *Let X and X' be two (pre)states of the pretableau and let $A \subseteq \Sigma$ be a coalition of agents. We say that X'* is A-reachable from *X if there exists a sequence of (pre)states $X = X_0, X_1, \ldots, X_m = X'$ such that for every $0 \leq i < m$, either $X_i \Longrightarrow X_{i+1}$ or $X_i \overset{\neg\mathbf{D}_{B_i}\varphi_i}{\longleftrightarrow} X_{i+1}$ for some $B_i \supseteq A$ and $\varphi_i \in \mathcal{L}$.*

If X' is $\emptyset$-reachable from X, we say that X is a predecessor of *X'.*

The immediate predecessor states of a given prestate are called its parents*; likewise for the immediate predecessor prestates of a given state.*

Definition 12. *Let X and X' be (pre)tates of a pretableau such that X' is A-reachable from X. A* defect with respect to *X and X' in the pretableau is a pair of formulas $(\mathbf{D}_A\psi, \neg\psi)$ such that $\mathbf{D}_A\psi \in X$ and $\neg\psi \in X'$ or vice versa.*

We are now in a position to state our construction rules. The first rule, **(SR)**, prescribes how to create states from prestates. Given a set $\Gamma \subseteq \mathcal{L}$, we say that Δ is a *minimal expansion of* Γ if Δ is (not necessarily fully) expanded, $\Gamma \subseteq \Delta$, and there is no Δ' such that $\Gamma \subseteq \Delta' \subset \Delta$ and Δ' is expanded.

Rule (SR) Given a prestate Γ, do the following:

1. Add all minimal expansions Δ of Γ as *states*;
2. for each so obtained state Δ, put $\Gamma \Longrightarrow \Delta$;
3. if, however, the pretableau already contains a state Δ' which is a predecessor of Γ, has the same label as Δ, and is such that adding an $\Longrightarrow$-edge from Γ to Δ' does not create any defect with respect to Γ and Δ', then do not create a state with a label Δ, but put $\Gamma \Longrightarrow \Delta'$.

We denote by $\mathbf{states}(\Gamma)$ the (finite) set $\{\, \Delta \mid \Gamma \Longrightarrow \Delta \,\}$.

The second construction rule, **(DR)**, prescribes how to create prestates from states. This rule does not apply to patently inconsistent states, as such states cannot be satisfied in any CMAEHS.

Rule (DR): Given a state Δ which is not patently inconsistent and $\neg\mathbf{D}_A\varphi \in \Delta$, do the following:

1. Create a new prestate[3] $\Gamma = \{\neg\varphi\} \cup \bigcup_{A' \subseteq A}\{\, \mathbf{D}_{A'}\psi \mid \mathbf{D}_{A'}\psi \in \Delta \,\} \cup \bigcup_{A' \subseteq A}\{\, \neg\mathbf{D}_{A'}\psi \mid \neg\mathbf{D}_{A'}\psi \in \Delta \text{ and } \neg\mathbf{D}_{A'}\psi \neq \neg\mathbf{D}_A\varphi \,\}$;
2. connect Δ to Γ with $\overset{\neg\mathbf{D}_A\varphi}{\longleftrightarrow}$;
3. if, however, the pretableau already contains a prestate Γ' which is a predecessor of Δ, has the same label as Γ, and is such that adding an $\overset{\neg\mathbf{D}_A\varphi}{\longleftrightarrow}$-edge from Δ to Γ' does not create any defect with respect to Δ and Γ', then do not add a prestate with label Γ, but simply connect Δ to Γ' with $\overset{\neg\mathbf{D}_A\varphi}{\longleftrightarrow}$.

Notice that the rules **(SR)** and **(DR)** have two built-in conditions for reusing states and prestates. First, we only reuse the predecessors of a given (pre)state, which amounts to looping back to the same branch of the pretableau tree—the reuse of (pre)states from other branches is not allowed. Secondly, we only loop back when this does not create a defect in the pretableau. These two conditions are needed to make our procedure sound. Notice that there no conditions on reuse of (pre)states in the tableau procedures in [12] and [4].

[3] This clause propagates the subformulae $\neg\mathbf{D}_{A'}\psi$ without this being absolutely necessary. The rule can be optimized, but in its present form it significantly simplifies proving completeness of the procedure.

When building a tableau for $\theta \in \mathcal{L}$, the construction phase begins with creating a single prestate $\{\theta\}$. Afterwards, we alternate between **(SR)** and **(DR)**: first, **(SR)** is applied to the prestates created at the previous stage of the construction, then **(DR)** is applied to the states created at the previous stage. The construction phase comes to an end when every prestate required to be added to the pretableau has already been added (as prescribed in point 3 of **(SR)**), or when we end up with states to which **(DR)** does not apply.

5.3 Termination of the Construction Phase

To show that the construction stage terminates, we use the concept of an extended closure of a formula.

Definition 13. *Let $\theta \in \mathcal{L}$. The* closure *of θ is the least set of formulae $\mathsf{cl}(\theta)$ containing θ, closed under subformulae, and satisfying the following condition: if $\mathbf{C}_A\varphi \in \mathsf{cl}(\theta)$, then $\mathbf{D}_a(\varphi \wedge \mathbf{C}_A\varphi) \in \mathsf{cl}(\theta)$ for every $a \in A$.*

The extended closure *of θ, denoted $\mathsf{ecl}(\theta)$, is the least set such that $\varphi, \neg\varphi \in \mathsf{ecl}(\theta)$ for every $\varphi \in \mathsf{cl}(\theta)$.*

Proposition 1. *The construction of $\mathcal{P}^\theta$ terminates for every formula θ.*

Proof sketch. Clearly, $\mathsf{ecl}(\theta)$ is finite. Therefore, the set of states and prestates of $\mathcal{P}^\theta$ is finite since they are all subsets of $\mathsf{ecl}(\theta)$. Then, to prove that $\mathcal{P}^\theta$ is finite and, hence, the construction phase terminates, it is suffices to show that on every branch of $\mathcal{P}^\theta$, we reuse each (pre)state only finitely many times. This will imply that every branch of $\mathcal{P}^\theta$ is finite and hence, by König's lemma, $\mathcal{P}^\theta$ itself is finite.

We only consider the case of reusing prestates; the case of reusing states is very similar. Suppose that a state Δ needs an A-successor prestate and has a predecessor prestate Γ with the same label. Suppose that reusing Γ would create a (not previously existing) defect $(\mathbf{D}_B\psi, \neg\psi)$ with respect to Γ and Δ. This is only possible if Δ is C-reachable from Γ for some coalition C such that $B \subseteq A \cup C$, while $B \not\subseteq A$ (otherwise Γ would not be consistent and not $B \not\subseteq C$ (otherwise, the defect is already there). Then, instead of reusing Γ, the rule **(DR)** creates a new prestate Γ' with the same label as Γ, which is an A-successor of Δ. But then, next time when a successor prestate with the label of Γ is needed further down the same branch, reusing Γ would be safe, because the A-transition from Δ to Γ' breaks the path B-path from Γ to the current prestate, thus preventing the same defect from occurring. Thus, the number of defects that can arise as a result of reusing prestates has decreased, so eventually no defects will be possible as a result of reusing prestates on the given branch, hence the creation of new prestates on this branch will come to an end. □

5.4 Prestate Elimination Phase

At this phase, we remove from $\mathcal{P}^\theta$ all the prestates and unmarked arrows, by applying the following rule (the resultant graph is denoted $\mathcal{T}_0^\theta$ and called the *initial tableau*):

(PR) For every prestate Γ in $\mathcal{P}^\theta$, do the following:

1. Remove Γ from $\mathcal{P}^\theta$;
2. If there is a state Δ in $\mathcal{P}^\theta$ with $\Delta \xleftrightarrow{\chi} \Gamma$, then for every state $\Delta' \in \mathbf{states}(\Gamma)$, put $\Delta \xleftrightarrow{\chi} \Delta'$;

5.5 State Elimination Phase

During this phase, we remove from $\mathcal{T}_0^\theta$ states that are not satisfiable in any CMAEHS. Recall, that there are three reasons why a state Δ of $\mathcal{T}_0^\theta$ can turn out to be unsatisfiable: Δ is patently inconsistent, *or* satisfiability of Δ requires satisfiability of some other unsatisfiable "successor" states, *or* Δ contains an eventuality that is not realized in the tableau. Accordingly, we have three elimination rules, **(E1)**–**(E3)**.

Formally, the state elimination phase is divided into stages; we start at stage 0 with $\mathcal{T}_0^\theta$; at stage $n+1$, we remove from the tableau $\mathcal{T}_n^\theta$ obtained at the previous stage exactly one state, by applying one of the elimination rules, thus obtaining the tableau $\mathcal{T}_{n+1}^\theta$. We state the rules below, where S_m^θ denotes the set of states of $\mathcal{T}_m^\theta$.

(E1) If $\{\varphi, \neg\varphi\} \subseteq \Delta \in S_n^\theta$, then obtain $\mathcal{T}_{n+1}^\theta$ by eliminating Δ from $\mathcal{T}_n^\theta$.

(E2) If Δ contains a formula $\chi = \neg\mathbf{D}_A\varphi$ and all states reachable from Δ by single arrows marked with χ have been eliminated at previous stages, obtain $\mathcal{T}_{n+1}^\theta$ by eliminating Δ from $\mathcal{T}_n^\theta$.

For the third elimination rule, we need the concept of *eventuality realization*. We say that the eventuality $\xi = \neg\mathbf{C}_A\varphi$ is realized at Δ in $\mathcal{T}_n^\theta$ if either $\neg\varphi \in \Delta$ or there exists in $\mathcal{T}_n^\theta$ a finite path $\Delta_0, \Delta_1, \ldots, \Delta_m$ such that $\Delta_0 = \Delta$, $\neg\varphi \in \Delta_m$, and for every $0 \leq i < m$ there exist $\chi_i = \mathbf{D}_{B_i}\psi_i$ such that $B_i \subseteq A$ and $\Delta_i \xleftrightarrow{\chi_i} \Delta_{i+1}$.

(E3) If $\Delta \in S_n^\theta$ contains an eventuality $\neg\mathbf{C}_A\varphi$ that is not realized at Δ in $\mathcal{T}_n^\theta$, then obtain $\mathcal{T}_{n+1}^\theta$ by removing Δ from $\mathcal{T}_n^\theta$.

We check for realization of ξ by running the following procedure that marks all states that realize an eventuality ξ in $\mathcal{T}_n^\theta$. Initially, we mark all $\Delta \in S_n^\theta$ such that $\neg\varphi \in \Delta$. Then, we repeatedly do the following: if $\Delta \in S_n^\theta$ is unmarked and there exists at least one Δ' such that $\Delta \xleftrightarrow{\mathbf{D}_B\psi} \Delta'$ for some $B \subseteq A$ and $\psi \in \mathcal{L}$ and Δ' is marked, then Δ gets marked. The procedure ends when no more states get marked. Note that marking is carried out with respect to a fixed eventuality ξ and is, therefore, repeated as many times as the number of eventualities in (the states) of a tableau.

We have so far described individual rules; to describe the state elimination phase as a whole, we must specify the order of their application. First, we apply **(E1)** to all the states of $\mathcal{T}_0^\theta$; once this is done, we do not need to apply **(E1)** again. The cases of **(E2)** and **(E3)** are more involved. After having applied **(E3)** we could have removed all the states accessible from some Δ along the arrows marked with some formula χ; hence, we need to reapply **(E2)** to the resultant tableau to remove such Δ's. Conversely, after having applied **(E2)**,

we could have thrown away some states that were needed for realizing certain eventualities; hence, we need to reapply **(E3)**. Therefore, we need to apply **(E3)** and **(E2)** in a dovetailed sequence that cycles through all the eventualities. More precisely, we arrange all eventualities occurring in the tableau obtained from $\mathcal{T}_0^\theta$ after having applied **(E1)** in a list: $\xi_1, \ldots, \xi_m$. Then, we proceed in cycles. Each cycle consists of alternatingly applying **(E3)** to the pending eventuality (starting with ξ_1), and then applying **(E2)** to the resulting tableau, until all the eventualities have been dealt with. These cycles are repeated until no state is removed in a whole cycle. Then, the state elimination phase is over.

The graph produced at the end of the state elimination phase is called the *final tableau for* θ, denoted by $\mathcal{T}^\theta$, and its set of states is denoted by S^θ.

Definition 14. *The final tableau* $\mathcal{T}^\theta$ *is* open *if* $\theta \in \Delta$ *for some* $\Delta \in S^\theta$*; otherwise,* $\mathcal{T}^\theta$ *is* closed.

The tableau procedure returns "no" if the final tableau is closed; otherwise, it returns "yes" and, moreover, provides sufficient information for producing a finite model satisfying θ; that construction is sketched in Section 6.

6 Soundness, Completeness, and Complexity

The *soundness* of a tableau procedure amounts to claiming that if the input formula θ is satisfiable, then the tableau for θ is open. To establish soundness of the overall procedure, we use a series of lemmas showing that every rule by itself is sound; the soundness of the overall procedure is then an easy consequence. The proofs of the following two lemmas are straightforward.

Lemma 4. *Let* Γ *be a prestate of* $\mathcal{P}^\theta$ *such that* $\mathcal{M}, s \Vdash \Gamma$ *for some CMAEM* $\mathcal{M}$ *and* $s \in \mathcal{M}$*. Then,* $\mathcal{M}, s \Vdash \Delta$ *holds for at least one* $\Delta \in \mathbf{states}(\Gamma)$*.*

Lemma 5. *Let* $\Delta \in S_0^\theta$ *be such that* $\mathcal{M}, s \Vdash \Delta$ *for some CMAEM* $\mathcal{M}$ *and* $s \in \mathcal{M}$*, and let* $\neg\mathbf{D}_A\varphi \in \Delta$*. Then, there exists* $t \in \mathcal{M}$ *such that* $(s,t) \in \mathcal{R}_A^D$ *and* $\mathcal{M}, t \Vdash \{\neg\varphi\} \cup \bigcup_{A' \subseteq A}\{\, \mathbf{D}_{A'}\psi \mid \mathbf{D}_{A'}\psi \in \Delta \,\} \cup \bigcup_{A' \subseteq A}\{\, \neg\mathbf{D}_{A'}\psi \mid \neg\mathbf{D}_{A'}\psi \in \Delta$ *and* $\neg\mathbf{D}_{A'}\psi \neq \neg\mathbf{D}_A\varphi \,\}$*.*

Lemma 6. *Let* $\Delta \in S_0^\theta$ *be such that* $\mathcal{M}, s \Vdash \Delta$ *for some CMAEM* $\mathcal{M}$ *and* $s \in \mathcal{M}$*, and let* $\neg\mathbf{C}_A\varphi \in \Delta$*. Then,* $\neg\mathbf{C}_A\varphi$ *is realized at* Δ *in* $\mathcal{T}_n^\theta$*.*

Proof idea. As $\neg\mathbf{C}_A\varphi$ is true at s, there is a path in $\mathcal{M}$ from s leading to a state satisfying $\neg\varphi$. As the tableaux organize the exhaustive search, a chain of tableau states corresponding to those states in the model will be produced. □

Theorem 3. *If* $\theta \in \mathcal{L}$ *is satisfiable in a CMAEM, then* $\mathcal{T}^\theta$ *is open.*

Proof sketch. Using the preceding lemmas, show by induction on the number of stages in the state elimination process that no satisfiable state can be eliminated due to **(E1)**–**(E3)**. The claim then follows from Lemma 4. □

The *completeness* of a tableau procedure means that if the tableau for a formula θ is open, then θ is satisfiable in a CMAEM. In view of Theorem 2, it suffices to show that an open tableau for θ can be turned into a CMAEHS for θ.

Lemma 7. *If $\mathcal{T}^\theta$ is open, then there exists a CMAEHS for θ.*

Proof sketch. The CMAEHS $\mathcal{H}$ for θ is built out of the so-called *final tree components*. Each component is a tree-like CMAES with nodes labeled with states from S^θ, and is associated with a state $\Delta \in S^\theta$ and an eventuality $\xi \in \mathsf{cl}(\theta)$ (such a component is denoted by $T_{\Delta,\xi}$). If $\xi \notin \Delta$, then $T_{\Delta,\xi}$ is a simple tree, whose root is labeled with Δ, that has exactly one leaf associated with each formula $\neg\mathbf{D}_A\psi$ marking an arrow from Δ to some $\Delta' \in S^\theta$; this leaf is labeled by Δ' and connected to the root by relation $\mathcal{R}^D_A$. If $\xi \in \Delta$, take the chain realizing χ at Δ and give each node "enough" successors, as prescribed above for simple trees. The crucial fact is that if ξ' is an eventuality in Δ that is not "realized" inside $T_{\Delta,\xi}$, then ξ' belongs to every leaf of $T_{\Delta,\xi}$. This allows us to stitch up all the $T_{\Delta,\xi}$'s into a Hintikka structure. The procedure is recursive. All the eventualities are queued. We start from the component uniquely associated with θ (say, we take $T_{\Delta,\theta}$ where Δ is the least numbered state containing θ; such a state exists as the tableau is open) and then replace each leaf of the structure built so far with the component associated with the set marking the leaf and the pending eventuality. The procedure is repeated in cycles until we have attached enough components to realize all eventualities. To obtain a CMAEHS, we put $H(\Delta) = \widehat{\Delta}$ for all Δ's, where $\widehat{\Delta}$ denotes the full expansion of Δ. □

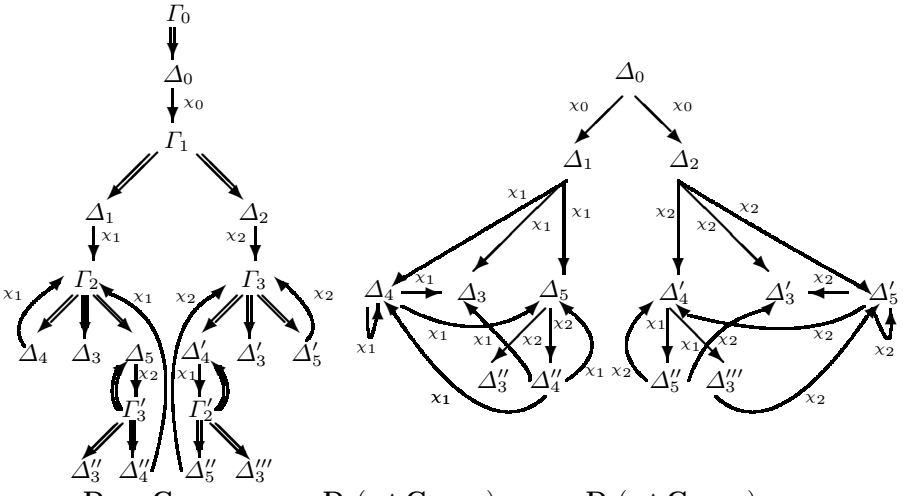
Theorem 4 (Completeness). *Let $\theta \in \mathcal{L}$ and let $\mathcal{T}^\theta$ be open. Then, θ is satisfiable in a CMAEM.*

Proof. Immediate from Lemma 7 and Theorem 2. □

As for complexity of the procedure, for lack of space, we only state that our procedure runs in exponential time, which together with the lower bound from [8] implies that **CMAEL(CD)**-satisfiability is **ExpTime**-complete.

7 Example

Let $\theta = \neg\mathbf{D}_{\{a,c\}}\mathbf{C}_{\{a,b\}}p \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)$, where $\Sigma = \{a, b, c\}$. To save space, we replace θ by the set of its conjuncts $\Theta = \{\neg\mathbf{D}_{\{a,c\}}\mathbf{C}_{\{a,b\}}p, \mathbf{C}_{\{a,b\}}(p \wedge q)\}$. The picture on the left below represents the final pretableau for Θ, while the picture on the right represents the initial tableau. Under the pictures, we list formulae that occur in the labels of states and prestates.



$\chi_0 = \neg\mathbf{D}_{\{a,c\}}\mathbf{C}_{\{a,b\}}p$; $\chi_1 = \neg\mathbf{D}_a(p \wedge \mathbf{C}_{\{a,b\}}p)$; $\chi_2 = \neg\mathbf{D}_b(p \wedge \mathbf{C}_{\{a,b\}}p)$;
$\Gamma_0 = \{\neg\mathbf{D}_{\{a,c\}}\mathbf{C}_{\{a,b\}}p, \mathbf{C}_{\{a,b\}}(p \wedge q)\}$;
$\Delta_0 = \{\neg\mathbf{D}_{\{a,c\}}\mathbf{C}_{\{a,b\}}p, \mathbf{C}_{\{a,b\}}(p \wedge q), \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]$,
$\mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], (p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q), (p \wedge q), p, q\}$;
$\Gamma_1 = \{\neg\mathbf{C}_{\{a,b\}}p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Delta_1 = \{\neg\mathbf{C}_{\{a,b\}}p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], \neg\mathbf{D}_a(p \wedge \mathbf{C}_{\{a,b\}}p), (p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)$,
$(p \wedge q), p, q, \mathbf{C}_{\{a,b\}}(p \wedge q), \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Delta_2 = \{\neg\mathbf{C}_{\{a,b\}}p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], \neg\mathbf{D}_b(p \wedge \mathbf{C}_{\{a,b\}}p), (p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)$,
$(p \wedge q), p, q, \mathbf{C}_{\{a,b\}}(p \wedge q), \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Gamma_2 = \Gamma_2' = \{\neg(p \wedge \mathbf{C}_{\{a,b\}}p), \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Gamma_3 = \Gamma_3' = \{\neg(p \wedge \mathbf{C}_{\{a,b\}}p), \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Delta_3 = \Delta_3' = \Delta_3'' = \Delta_3''' = \{\neg p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], (p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)$,
$(p \wedge q), p, q, \mathbf{C}_{\{a,b\}}(p \wedge q), \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Delta_4 = \Delta_4' = \Delta_4'' = \{\neg\mathbf{C}_{\{a,b\}}p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], \neg\mathbf{D}_a(p \wedge \mathbf{C}_{\{a,b\}}p)$,
$(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q), (p \wedge q), p, q, \mathbf{C}_{\{a,b\}}(p \wedge q), \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$;
$\Delta_5 = \Delta_5' = \Delta_5'' = \{\neg\mathbf{C}_{\{a,b\}}p, \mathbf{D}_a[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)], \neg\mathbf{D}_b(p \wedge \mathbf{C}_{\{a,b\}}p)$,
$(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q), (p \wedge q), p, q, \mathbf{D}_b[(p \wedge q) \wedge \mathbf{C}_{\{a,b\}}(p \wedge q)]\}$.

During the state-elimination phase, states Δ_3, Δ_3', Δ_3'', and Δ_3''' are removed due to **(E1)**, as they contain a patent inconsistency $(p, \neg p)$. Then, states Δ_1, Δ_2, Δ_4, Δ_4', Δ_4'', Δ_5, Δ_5', and Δ_5'' are eliminated due to **(E3)**, as all of them contain the unrealized eventuality $\neg\mathbf{C}_{\{a,b\}}p$. Finally, Δ_0 gets eliminated due to **(E2)**, as it has lost all its successors along the arrow marked with χ_0. Thus, the final tableau for Θ is an empty graph; therefore, Θ is *unsatisfiable*.

8 Concluding Remarks

We have developed a sound and complete, incremental-tableau-based decision procedure for the full coalitional multiagent epistemic logic **CMAEL(CD)**. We are convinced that this style of tableaux is more intuitive, practically more efficient, and more flexible than the top-down tableaux, e.g., developed for a fragment of this logic in [8], and therefore suitable both for manual and automated

execution. In particular, it is amenable to extension with strategic abilities operators of the Alternating-time temporal logic **ATL**, tableaux for which were developed in [4]. Merging these two systems is a topic of our future work.

Acknowledgments

We gratefully acknowledge the financial support from the National Research Foundation of South Africa through a grant for the first author, and from the Claude Harris Leon Foundation, funding the second author's post-doctoral fellowship at the University of the Witwatersrand, during which this research was done. We also thank the referees for useful comments and references.

References

1. Dziubinski, M., Verbrugge, R., Dunin-Keplicz, B.: Complexity issues in multiagent logics. Fundamenta Infomaticae 75, 239–262 (2007)
2. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
3. Fagin, R., Halpern, J.Y., Vardi, M.Y.: What can machines know? On the properties of knowledge in distributed systems. Journal of the ACM 39(2), 328–376 (1992)
4. Goranko, V., Shkatov, D.: Tableau-based decision procedures for logics of strategic ability in multi-agent systems. ACM Transactions on Computational Logic (to appear), `http://tocl.acm.org/accepted.html`
5. Goranko, V., Shkatov, D.: Tableau-based decision procedure for the multi-agent epistemic logic with operators of common and distributed knowledge. In: Proc. of the 6th IEEE Iternational Conference on Software Engineering and Formal Methods. IEEE Press, Los Alamitos (2008), `http://arxiv.org/abs/0808.4133`
6. Halpern, J.Y.: Using reasoning about knowledge to analyze distributed systems. Annual Review of Computer Science 2, 37–68 (1987)
7. Halpern, J.Y., Moses, Y.: Knowledge and common knowledge in a distributed environment. Journal of ACM 37(3), 549–587 (1990)
8. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence 54, 319–379 (1992)
9. van der Hoek, W., Meyer, J.-J.C.: Making some issues of implicit knowledge explicit. International Journal of Foundations of Computer Science 3(2), 193–224 (1992)
10. van der Hoek, W., Meyer, J.-J.C.: Epistemic Logic for AI and Computer Science. In: CUP (1995)
11. van der Hoek, W., Meyer, J.-J.C.: A complete epistemic logic for multiple agents–combining distributed and common knowledge. In: Bacharach, M.O.L., et al. (eds.) Epistemic Logic and the Theory of Games and Decisions, pp. 35–68. Kluwer Academic Publishers, Dordrecht (1997)
12. Wolper, P.: The tableau method for temporal logic: an overview. Logique et Analyse 28(110-111), 119–136 (1985)

A Clausal Approach to Proof Analysis in Second-Order Logic*

Stefan Hetzl, Alexander Leitsch, Daniel Weller, and Bruno Woltzenlogel Paleo

Institute of Computer Languages (E185),
Vienna University of Technology,
Favoritenstraße 9, 1040 Vienna, Austria
{hetzl,leitsch,weller,bruno}@logic.at

Abstract. This work defines an extension CERES^2 of the first-order cut-elimination method CERES to the subclass of sequent calculus proofs in second-order logic using quantifier-free comprehension. This extension is motivated by the fact that cut-elimination can be used as a tool to extract information from real mathematical proofs, and often a crucial part of such proofs is the definition of sets by formulas. This is expressed by the comprehension axiom scheme, which is representable in second-order logic. At the core of CERES^2 lies the production of a set of clauses CL(φ) from a proof φ that is always unsatisfiable. From a resolution refutation γ of CL(φ), a proof without essential cuts can be constructed. The main theoretical obstacle in the extension of CERES to second-order logic is the construction of this proof from γ. This issue is solved for the subclass considered in this paper. Moreover, we discuss the problems that have to be solved to extend CERES^2 to the complete class of second-order proofs. Finally, the method is applied to a simple mathematical proof that involves induction and comprehension and the resulting proof is analyzed.

1 Introduction

The discipline of *proof mining* deals with the extraction of information from formal proofs. Different methods have been applied successfully (see [1], [2]). This work considers the approach of using (partial) cut-elimination to extract hidden information from proofs.

The first-order cut-elimination method CERES (cut-elimination by resolution) has several advantages over the traditional reductive cut-elimination methods: The main computational advantage is that the reductive methods are subsumed by CERES (i.e. every proof obtained by a reductive method can also be obtained by CERES, see [2]), and secondly, a non-elementary speed-up over Gentzen's method by the use of CERES is possible (see [3]). Another, purely proof-theoretic, advantage is that a CERES method for a proof system provides a strong regularity theorem on the structure of cut-free proofs: The result of

* Supported by the Austrian Science Fund (project P19875).

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 214–229, 2009.

applying CERES is a proof which is composed of instances of (otherwise unchanged) parts of the original proof. This knowledge about the cut-free proofs cannot be obtained by traditional reductive methods as they perturb the structure of the proof. The CERES method has been implemented in the `CERES` system[1]. In this paper, we present the extension of CERES to CERES^2, a cut-elimination method for second-order logic, which is based on a set of clauses that is extracted from a proof with cuts, the *characteristic clause set*.

The benefits of CERES^2 over traditional cut-elimination methods are twofold: Firstly, the characteristic clause set can be regarded as the kernel of the proof with cuts and as such can provide valuable information that a human could not easily read off of a formal proof (for some evidence supporting this, see [4] and [5]). Secondly, due to the use of a resolution calculus at the core of CERES^2, theoretical and practical advances in higher-order theorem proving may enhance the power of the method.

An inherent limitation of the CERES method (and indeed of all first-order cut-elimination procedures) is that proofs that use comprehension cannot be handled in a straightforward way, as comprehension is essentially a second-order property. In CERES^2, we will be able to handle such proofs in a natural way. The subclass of proofs we are considering here is the class of proofs where comprehension is restricted to quantifier-free formulas. This choice is motivated in part by the fact that converting a resolution refutation to a sequent calculus proof in the presence of arbitrary comprehension (and, therefore, skolemization) is problematic. Indeed, it turns out that, in CERES^2, the construction of proof projections is a highly complicated matter, in contrast to the first-order case.

Note that, due to lack of space, detailed proofs are not developed here, but can be found in [6].

2 The Second-Order Language

Here, we consider a monadic second-order logic based on Church's simply typed λ-calculus [7] and fix the set of *base types* $BT := \{\iota, o\}$, where ι denotes the type of individuals and o the boolean type. The set $\mathcal{T}$ of *types* is built in the usual inductive way over the BT. In contrast to the second-order logic as defined in e.g. [8], we include in the language more objects of order ≤ 2 to allow skolemization, although quantification is restricted to individuals and unary predicates on individuals (i.e. variables of types ι and $\iota \to o$).

We assume given a set of symbols S together with a function $\tau : S \mapsto \mathcal{T}$ assigning types to symbols, where S can be partitioned into the sets V (individual variables), CS (individual constants), FS^n (function symbols), PC^n (predicate constants), PV (unary predicate variables) s.t.

1. For all $x \in V$, $\tau(x) = \iota$,
2. for all $c \in CS$, $\tau(c) = \iota$,

[1] `http://www.logic.at/ceres`

3. for all $f \in FS^n$, $n \geq 1$: $\tau(f) = t_1 \to \ldots \to t_n \to \iota$ where for $1 \leq i \leq n$, $t_i = \iota$ or $t_i = \iota \to o$,
4. for all $P \in PC^n$, $n \geq 0$: $\tau(P) = t_1 \to \ldots \to t_n \to o$ where for $1 \leq i \leq n$, $t_i = \iota$ or $t_i = \iota \to o$,
5. for all $X \in PV$, $\tau(X) = \iota \to o$.

We additionally require that each member of this partition is countably infinite. We define $PC := \bigcup_{i \geq 0} PC^i$ and $FS := \bigcup_{i \geq 1} FS^i$. The set of *expressions* $\mathcal{E}$ is defined inductively in the usual way over the set of symbols together with the symbols $\neg, \wedge, \vee, \to, \exists, \forall, \lambda, ., (,)$ (keeping in mind the restriction on the order of the types and on the types of the quantified variables). We use infix notation for familiar function symbols and predicates (e.g. $+, =$).

Definition 1. *The* set of second-order formulas *or simply* formulas $SOF := \{F \mid F \in \mathcal{E}, \tau(F) = o\}$. *If* $F \in SOF$, $F \equiv P(t_1, \ldots, t_n)$, $P \in PC \cup PV$, *then* F *is called* atomic.

For atomic formulas $P(t_1, \ldots, t_n)$ we may also write $t_1 \in P(t_2, \ldots, t_n)$. We define the set of *lambda terms* $LT := \{t \mid t \in \mathcal{E}, t \equiv \lambda x.F, \tau(F) = o\}$ and the set of *terms* $T := \{t \mid t \in \mathcal{E}, \tau(t) = \iota\}$. *Polarity* of subexpressions w.r.t. formulas and sequents, *strong* and *weak* quantifiers, the *scope* of quantifiers, *closed* formulas, β-reduction are defined as usual. We assume a variable convention (i.e. variables are renamed appropriately to avoid conflicts).

As proof system for the input and output proofs of the CERES^2 method, we use the sequent calculus **LKDe²**. This calculus is based on **LK²** as defined in [9], which consists of the usual structural, propositional, and first-order rules together with second-order quantifier introduction rules that incorporate comprehension. **LKDe²** extends **LK²** by rules for first-order equality handling:

$$\frac{\Gamma \vdash \Delta, s = t \quad \Pi \vdash \Lambda, A[s]}{\Gamma, \Pi \vdash \Delta, \Lambda, A[t]} =: r_1 \qquad \frac{\Gamma \vdash \Delta, t = s \quad \Pi \vdash \Lambda, A[s]}{\Gamma, \Pi \vdash \Delta, \Lambda, A[t]} =: r_2$$

LKDe² also includes the rules $=: l_1$ and $=: l_2$, and rules for the introduction of definitions (for details, see [10]). All the rules in **LKDe²** are multiplicative. As axioms we allow the usual tautological sequents $A \vdash A$ for an atomic formula A as well as arbitrary atomic sequents without second-order variables (which is useful for conveniently axiomatizing a background theory). Additionally, if $\mathcal{C}$ is a set of atomic sequents, then we say that π is *an* **LKDe²***-proof from* $\mathcal{C}$ if for every initial sequent S of π, S is either an axiom, or S is in $\mathcal{C}$. As an intermediary calculus for the construction of a resolution refutation, we use the resolution calculus discussed in the next section.

3 The Second-Order Resolution Calculus

In this section, we briefly present the resolution calculus we will need for the CERES^2 method. Note that in second-order logic, in contrast to first-order logic, clauses are not closed under substitution, so the transformation of a formula to

clause form has to be incorporated into the calculus, instead of being used just in a preprocessing step.

To use a resolution calculus with CERES^2, it must be possible to use the resolution refutation of a particular set of clauses (the characteristic clause set, see Section 4) as the skeleton of an **LKDe^2**-proof that contains no non-atomic cuts. Intuitively, the following requirements arise:

1. Only literals (i.e. atomic formulas and their negations) may be resolved.
2. It must be possible to produce a propositional resolution refutation from instances of the refuted set of clauses.

Requirement 1 stems from the fact that CERES^2 is a cut-elimination method, and the resolution rule will be translated to the cut rule in **LKDe^2**. Requirement 2 is due to the fact that substitution is integrated in the resolution calculus, while this is not the case with **LKDe^2**.

The resolution calculus we are considering here is a restricted version of the higher-order resolution calculus defined by P.B. Andrews in [11].

Definition 2. *We define a* clause *as a sequent* $C := A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ *with* A_i, B_i *atomic.*

In this paper, the transformation to conjunctive normal form (CNF) is the standard transformation that preserves logical equivalence.

Definition 3. *Let* F *be a quantifier-free formula. Let, modulo commutativity and associativity of* $\vee$, $\mathrm{CNF}(F) \equiv (\neg A^1_1 \vee \ldots \vee \neg A^1_{k_1} \vee B^1_1 \vee \ldots \vee B^1_{l_1}) \wedge \ldots \wedge (\neg A^n_1 \vee \ldots \vee \neg A^n_{k_n} \vee B^n_1 \vee \ldots \vee B^n_{l_n})$. *For* $i \in \{1, \ldots, n\}$, *define the atomic sequent* $C_i \equiv A^i_1, \ldots, A^i_{k_i} \vdash B^i_1, \ldots, B^i_{l_i}$. *Then the* clause form *of* F *is defined as the set* $\{C_1, \ldots, C_n\}$.

Let $S \equiv F_1, \ldots, F_n \vdash G_1, \ldots, G_m$ *be a quantifier-free sequent, then the clause form of* S *is defined as the clause form of* $(F_1 \wedge \ldots \wedge F_n) \rightarrow (G_1 \vee \ldots \vee G_m)$.

A substitution is a pair of mappings: The first maps variables to terms, while the second maps predicate variables to lambda terms. The result of the application of a substitution σ to an expression e is e after replacing all variables by the respective terms and all predicate variables by the respective lambda terms and reducing to β-normal form, this will be denoted by $e\sigma$. A substitution is called quantifier-free if all the (lambda-)terms are quantifier-free.

Definition 4. *We define the application of a quantifier-free substitution* σ *to a set of clauses* $\mathcal{C} = \{C_1, \ldots, C_n\}$, *denoted* $\mathcal{S}(\mathcal{C}, \sigma)$, *as the clause form of the set of quantifier-free sequents* $\{C_1\sigma, \ldots, C_n\sigma\}$. *Note that this includes transformation to CNF, therefore* $|\mathcal{S}(\mathcal{C}, \sigma)| \geq |\mathcal{C}|$.

With this definition, we can state the rules of our resolution calculus.

Definition 5. *In the following,* C, D *are clauses.*

1. C *is called* instance *of* D *if there exists a quantifier-free substitution* σ *s.t.* $C \in \mathcal{S}(\{D\}, \sigma)$.

2. *C is called* p-reduct *of D if C is D after omission of some multiply occuring atomic formulas on either side of the sequent.*
3. *Let L be an atom formula, $C \equiv \Gamma, L \vdash \Delta$ and $D \equiv \Gamma' \vdash L, \Delta'$, then the clause $\Gamma, \Gamma' \vdash \Delta, \Delta'$ is called a* resolvent *of $\{C, D\}$.*

Note that we defined resolution without the principle of most general unification (mgu). While the mgu-principle is vital to proof search, the proof transformations in CERES and CERES^2 require resolution proofs after application of global unifiers. Of course, this definition does not exclude the use of mgu-based provers in the phase of proof search.

Additionally, we use paramodulation rules that allow equality reasoning on the term level. The paramodulation rules are just the restrictions of the equational rules of $\mathbf{LKDe^2}$ to atomic sequents. With this, we can define the notion of a deduction in this calculus:

Definition 6. *Let $\mathcal{C}$ be a set of clauses and let C be a clause. A sequence $C_1, \ldots, C_n$ is called an* R-deduction *of C from $\mathcal{C}$ if it fulfills the following conditions: $C_n \equiv C$ and for all $i = 1, \ldots, n$:*

- *$C_i \in \mathcal{C}$ or*
- *C_i is an instance or a p-reduct of C_j for some $j < i$ or*
- *C_i is a resolvent of $\{C_j, C_k\}$ for some $j, k < i$ or*
- *C_i is the result of paramodulation of $\{C_j, C_k\}$ for $j, k < i$.*

An R-deduction of the empty sequent $\vdash$ from $\mathcal{C}$ is called an R-refutation of $\mathcal{C}$.

Finally, we state some lemmas that show that R-deductions can be transformed to $\mathbf{LKDe^2}$-proofs. These will be useful for showing the effectiveness of the CERES^2 method in the next section.

Lemma 1. *Let $C \equiv \Gamma \vdash \Delta$ be a clause, $\mathcal{D}$ be a set of clauses, ψ be a $\mathbf{LKDe^2}$-proof of $\Gamma, \Pi \vdash \Lambda, \Delta$ from $\mathcal{D}$ with only quantifier-free cuts, let σ be a quantifier-free substitution whose domain contains no variable which occurs free in $\Pi \cup \Lambda$ and let $\Gamma^* \vdash \Delta^* \in \mathcal{S}(\{C\}, \sigma)$. Then we can construct an $\mathbf{LKDe^2}$-proof ψ^* of $\Gamma^*, \Pi \vdash \Lambda, \Delta^*$ from $\mathcal{S}(\mathcal{D}, \sigma)$ with only quantifier-free cuts and with $|\psi^*| \leq |\psi| + \rho(|\Gamma\sigma \vdash \Delta\sigma|)$, where ρ is exponential if σ substitutes for a predicate variable in $\mathcal{D}$, and polynomial otherwise.*

Proof. By simulating the conjunctive normal form transformation in $\mathbf{LKDe^2}$ using cuts. For a complete proof, see [6].

Note that this result is weaker than the corresponding result in first-order logic: the proofs constructed in that setting do not contain any cuts. Still, as our interest is the extraction of information, this result suffices, as quantifier-free cuts do not contain interesting mathematical information.

Lemma 2. *Let R be an R-deduction of $\Gamma \vdash \Delta$ from a set of clauses $\mathcal{C}$. Then there exists an $\mathbf{LKDe^2}$-proof ψ of $\Gamma \vdash \Delta$ from $\mathcal{D}$ containing quantifier-free cuts only, where $\mathcal{D} = \{D \mid D \in \mathcal{S}(\mathcal{C}, \sigma)$ for some quantifier-free $\sigma\}$.*

Proof. By using Lemma 1 to replace instantiations, replacing resolution with cut, replacing paramodulation by the equality rules, and replacing p-reducts by contractions. For a full proof, see [6].

4 The CERES^2 Cut-Elimination Method

We now define the CERES^2 method, which will turn out to be a cut-elimination method for **LKDe^2**-proofs using quantifier-free comprehension.

Definition 7. *Let (R) be a weak second-order quantifier rule*

$$\frac{A\{X \leftarrow \lambda x.F\}, \Gamma \vdash \Delta}{(\forall X)A, \Gamma \vdash \Delta}\ \forall^2 : l \qquad \frac{\Gamma \vdash \Delta, A\{X \leftarrow \lambda x.F\}}{\Gamma \vdash \Delta, (\exists X)A}\ \exists^2 : r$$

then (R) is called quantifier-free if F does not contain quantifiers. We call an **LKDe^2***-proof π a* **QFC***-proof if all its weak second-order quantifier rule applications are quantifier-free.*

Note that as we allow non-tautological axioms, it is not in general possible to eliminate all cuts. This leads to the following notion: An **LKDe^2**-proof π is called in *atomic cut normal form (ACNF)* if all cut-formulas of π are atomic. An important technical tool in the CERES^2 method is the skolemization of proofs: this transformation removes strong quantifier rules from proofs and replaces the respective variables by Skolem terms.

Definition 8. *Let ψ be an* **LKDe^2***-proof. If the active formulas of all strong quantifier rules in ψ are ancestors of cut-formulas, then ψ is said to be in* Skolem form*.*

The following proposition shows that from a **QFC**-proof, we can indeed obtain a proof in Skolem form. Proofs in Skolem form allow the definition of proof projections by leaving out rules from the proof, as no eigenvariable violations can occur by doing so. This will be necessary to construct sound proofs in Definition 10. We use the structural skolemization operator sk on formulas and sequents, where sk replaces the strongly quantified variables by Skolem terms and drops the corresponding quantifiers (see [12]).

Proposition 1. *For every* **QFC***-proof ψ of S there exists a* **QFC***-proof ψ' of* $\mathrm{sk}(S)$ *in Skolem form.*

Proof. We obtain ψ' from ψ by dropping the strong quantifier rules going into the end-sequent and, on the path to the end-sequent, replacing the strongly quantified variables by the respective Skolem terms. For example, if S contains a positive occurrence α of $(\forall X)A(X)$ and the premise of the $\forall^2 : r$ introducing this quantifier is $\Pi \vdash \Lambda, A'(\Theta)$ and $t_1, \ldots, t_n$ are the (lambda-)terms eliminated by introductions of weak quantifiers dominating α and the corresponding skolem term in $\mathrm{sk}(S)$ is $\lambda z.P(z, x_1, \ldots, x_n)$, then we remove the $\forall^2 : r$ rule, replace its premise by $\Pi \vdash \Lambda, A'(\lambda z.P(z, t_1, \ldots, t_n))$, and modify the path to the end-sequent so that the occurrences of $t_1, \ldots, t_n$ in the Skolem term are eliminated by the weak quantifier rules. For a full proof, see [6].

Note that in this context, skolemization indeed does preserve validity (in contrast to what is observed in [13]), because the proposition we just stated generates a

proof of the skolemized formula from a proof of the unskolemized formula. As **LKDe²** is sound, the transformation is validity preserving. We now define some notation that will be useful in describing CERES^2.

Definition 9. *Let ρ be a unary rule, σ a binary rule, ψ, χ* **QFC***-proofs, then $\rho(\psi)$ is the* **QFC***-proof obtained by applying ρ to the end-sequent of ψ, and $\sigma(\psi, \chi)$ is the proof obtained from the proofs ψ and χ by applying σ.*

Let P, Q be sets of **QFC***-proofs. Then $P^{\Gamma \vdash \Delta} := \{\psi^{\Gamma \vdash \Delta} \mid \psi \in P\}$, where $\psi^{\Gamma \vdash \Delta}$ is ψ followed by weakenings adding $\Gamma \vdash \Delta$, and $P \times_\sigma Q := \{\sigma(\psi, \chi) \mid \psi \in P, \chi \in Q\}$.*

Let $C = \{\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_m \vdash \Delta_m\}$, $D = \{\Pi_1 \vdash \Lambda_1, \ldots, \Pi_n \vdash \Lambda_n\}$ be sets of clauses, then $C \times D := \{\Gamma_i, \Pi_j \vdash \Delta_i, \Lambda_j \mid i \leq m, j \leq n\}$.

We can now define the main parts of the CERES^2-method: the characteristic clause set and the set of proof projections of a proof π. The former will be always unsatisfiable and give rise to a resolution refutation, while the latter will allow the resolution refutation to be transformed into a proof of the end-sequent of π.

Definition 10. *Let π be a* **QFC***-proof in Skolem form. For each rule ρ in π, we define a set of cut-free* **QFC***-proofs, the set of* projections $\mathcal{P}_\rho(\pi)$ *of π, and a set of clauses, the* characteristic clause set $\mathrm{CL}_\rho(\pi)$ *of π, at the position of ρ.*

- *If ρ corresponds to an initial sequent, let $\Gamma_1 \vdash \Delta_1$ be the part of it which consists of ancestors of cut formulas, let $\Gamma_2 \vdash \Delta_2$ be the part which consists of ancestors of the end-sequent of π and define*

$$\begin{aligned} \mathcal{P}_\rho(\pi) &:= \{\Gamma_1, \Gamma_2 \vdash \Delta_2, \Delta_1\} \\ \mathrm{CL}_\rho(\pi) &:= \{\Gamma_1 \vdash \Delta_1\}. \end{aligned}$$

- *If ρ is a unary rule with immediate predecessor ρ' with $\mathcal{P}_{\rho'}(\pi) = \{\psi_1, \ldots, \psi_n\}$, distinguish:*
  - *(a) The active formulas of ρ are ancestors of cut formulas. Then*

$$\mathcal{P}_\rho(\pi) := \mathcal{P}_{\rho'}(\pi)$$

  - *(b) The active formulas of ρ are ancestors of the end-sequent. Then*

$$\mathcal{P}_\rho(\pi) := \{\rho(\psi_1), \ldots, \rho(\psi_n)\}$$

    Note that by assumption, all strong quantifier rules go into cuts, so ρ cannot be a strong quantifier rule, so no eigenvariable violation can occur here.

  In any case, $\mathrm{CL}_\rho(\pi) := \mathrm{CL}_{\rho'}(\pi)$.
- *Let ρ be a binary rule with immediate predecessors ρ_1 and ρ_2.*
  - *(a) If the active formulas of ρ are ancestors of cut-formulas, let $\Gamma_i \vdash \Delta_i$ be the ancestors of the end-sequent in the conclusion sequent of ρ_i and define*

$$\mathcal{P}_\rho(\pi) := \mathcal{P}_{\rho_1}(\pi)^{\Gamma_2 \vdash \Delta_2} \cup \mathcal{P}_{\rho_2}(\pi)^{\Gamma_1 \vdash \Delta_1}$$

    For the characteristic clause set, define

$$\mathrm{CL}_\rho(\pi) := \mathrm{CL}_{\rho_1}(\pi) \cup \mathrm{CL}_{\rho_2}(\pi)$$

(b) If the active formulas of ρ are ancestors of the end-sequent, then

$$\mathcal{P}_\rho(\pi) := \mathcal{P}_{\rho_1}(\pi) \times_\rho \mathcal{P}_{\rho_2}(\pi).$$

For the characteristic clause set, define

$$\mathrm{CL}_\rho(\pi) := \mathrm{CL}_{\rho_1}(\pi) \times \mathrm{CL}_{\rho_2}(\pi)$$

The set of projections *of π, $\mathcal{P}(\pi)$ is defined as $\mathcal{P}_{\rho_0}(\pi)$, and the* characteristic clause set *of π, $\mathrm{CL}(\pi)$ is defined as $\mathrm{CL}_{\rho_0}(\pi)$, where ρ_0 is the last rule of π.*

Example 1. Consider the proof ψ:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \in \Theta \vdash a \in \Theta}{\vdash a \in \Theta, a \notin \Theta}\neg : r \quad \dfrac{b \in \Theta \vdash b \in \Theta}{b \notin \Theta, b \in \Theta \vdash}\neg : l}{b \in \Theta, a \notin \Theta \to b \notin \Theta \vdash a \in \Theta}\to : l}{b \in \Theta, (\forall X)(a \in X \to b \in X) \vdash a \in \Theta}\forall^2 : l}{(\forall X)(a \in X \to b \in X) \vdash b \in \Theta \to a \in \Theta}\to : r}{(\forall X)(a \in X \to b \in X) \vdash (\forall X)(b \in X \to a \in X)}\forall^2 : r \quad \dfrac{\dfrac{\dfrac{\dfrac{b \in P \vdash b \in P \qquad a \in P \vdash a \in P}{b \in P \to a \in P, b \in P \vdash a \in P}\to : l}{b \in P \to a \in P \vdash b \in P \to a \in P}\to : r}{(\forall X)(b \in X \to a \in X) \vdash b \in P \to a \in P}\forall^2 : l}{(\forall X)(a \in X \to b \in X) \vdash b \in P \to a \in P}\,cut$$

where X, Θ are predicate variables, a, b are individual constants, P is a predicate constant, and the lambda terms used in the $\forall^2 : l$ rules are $\lambda x.x \notin \Theta$ and $\lambda x.x \in P$. Then

$$\begin{aligned}\mathrm{CL}(\psi) &= (\{\vdash a \in \Theta\} \times \{b \in \Theta \vdash\}) \cup \{\vdash b \in P\} \cup \{a \in P \vdash\} \\ &= \{b \in \Theta \vdash a \in \Theta;\ \vdash b \in P;\ \ a \in P \vdash\}\end{aligned}$$

and $\mathcal{P}(\psi)$ contains, among others, the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a \in \Theta \vdash a \in \Theta}{\vdash a \in \Theta, a \notin \Theta}\neg : r \quad \dfrac{b \in \Theta \vdash b \in \Theta}{b \notin \Theta, b \in \Theta \vdash}\neg : l}{b \in \Theta, a \notin \Theta \to b \notin \Theta \vdash a \in \Theta}\to : l}{b \in \Theta, (\forall X)(a \in X \to b \in X) \vdash a \in \Theta}\forall^2 : l}{b \in \Theta, (\forall X)(a \in X \to b \in X) \vdash b \in P \to a \in P, a \in \Theta}w : r$$

Note that for the soundness of Definition 10, we need the assumption that π is in Skolem form: if this were not the case, violations of eigenvariable conditions could appear in the projections. We will now prove the main properties of CERES^2.

Lemma 3. *Let π be a* **QFC***-proof in Skolem form. Then there exists an R-refutation of* $\mathrm{CL}(\pi)$.

Proof. Analogous to the proof of unsatisfiability of $\mathrm{CL}(\pi)$ for first-order logic in [3] by removing all rules of π except the ancestors of the cuts, and removing all formula occurrences in π except the ancestors of cuts, we construct a **QFC**-proof ψ of $\vdash$ from $\mathrm{CL}(\pi)$.

As we know from e.g. [14], reductive cut-elimination in second-order logic terminates, so we can apply it to ψ to eliminate all non-atomic cuts and obtain a proof ψ' of $\vdash$. First, note that ψ' consists of atomic cut, contraction and

permutation: weakening is automatically eliminated by cut-elimination. Denote the set of initial sequents of a proof φ by $init(\varphi)$. We will show that ψ' can be transformed into a proof ψ^* s.t. $init(\psi^*)$ consists of quantifier-free instances of clauses in $\mathcal{C}$. We can then take $\mathcal{D}$ as $init(\psi^*)$. We proceed by induction on the cut-elimination of ψ to obtain ψ'. As induction invariant, we take the following: ψ' can be transformed into a **QFC**-proof ψ^* s.t. $init(\psi^*)$ consists of quantifier-free instances of clauses in $\mathcal{C}$. For the base case, we take $\psi^* = \psi$, so as $init(\psi) = init(\psi^*)$ and ψ uses quantifier free comprehension, the invariant holds.

1. The cut-eliminiation performs a rank reduction on ψ. Then the initial sequents of ψ and ψ' coincide, except when performing rank reduction over a contraction: Here, we perform adequate renamings of eigenvariables in ψ' to keep regularity and take $\psi^* = \psi'$. Clearly, $init(\psi^*)$ consists of $init(\psi)$ together with some renamed variants of clauses in $init(\psi)$, and the lambda terms of the weak second-order quantifier rules are not changed, so the invariant holds.
2. The cut-elimination performs a grade reduction on ψ. The most interesting subcase is: The grade reduction is performed on second-order quantifier rules. Let $\sigma = \{X \leftarrow \lambda x.F\}$ be the substitution that is applied by the cut-elimination. By (IH), σ is quantifier-free. Let $init(\psi) = \{\Gamma_1 \vdash \Delta_1, \ldots, \Gamma_n \vdash \Delta_n\}$. Then
$$init(\psi') = \{(\Gamma_1 \vdash \Delta_1)\sigma, \ldots, (\Gamma_n \vdash \Delta_n)\sigma\}$$
is a set of propositional sequents, so for $1 \leq i \leq n$, we have cut-free proofs φ_i of $(\Gamma_i \vdash \Delta_i)\sigma$ from $\mathcal{S}(\{\Gamma_i \vdash \Delta_i\}, \sigma)$. Then take ψ^* to be ψ' where the leafs are replaced by the respective φ_i, then
$$init(\psi) = \mathcal{S}(\{\Gamma_1 \vdash \Delta_1\}, \sigma) \cup \ldots \cup \mathcal{S}(\{\Gamma_n \vdash \Delta_n\}, \sigma)$$
and the first part of the invariant holds. For the second part, note that as σ is quantifier-free and no new second-order quantifier rules are introduced in this step, all second-order quantifier rules are still quantifier-free.

ψ^* readily gives rise to an R-refutation of $\mathrm{CL}(\pi)$: First, derive the necessary instances of clauses in $\mathrm{CL}(\pi)$ used as leaves of ψ^* using instantiation, then, whenever atomic cuts are used in ψ^*, apply resolution, and whenever contractions are used in ψ^*, apply p-reduction.

Note that this lemma is just a theoretical tool to show the existence of a suitable refutation — in practice, the reductive methods used in the proof of the lemma are not used (as can be seen in the analysis of the example in Section 5).

We are now ready to define the CERES^2 method and state our central result.

Definition 11. *Let π be a* **QFC***-proof of S. Then the* CERES^2 *method is the following algorithm:*

1. *Compute a* **QFC***-proof π_{sk} of* $\mathrm{sk}(S)$.
2. *Compute* $\mathrm{CL}(\pi_{sk})$, $\mathcal{P}(\pi_{sk})$.

3. *Compute an R-refutation* γ *of* $\mathrm{CL}(\pi_{sk})$.
4. *Convert* γ *into an* $\mathbf{LKDe^2}$*-proof* γ' *of* $\vdash$ *from* $\mathrm{CL}(\pi_{sk})$.
5. *Plug instances of the proofs in* $\mathcal{P}(\pi_{sk})$ *into the leaves of* γ' *to obtain a proof* ψ *of* $\mathrm{sk}(S)$ *containing quantifier-free cuts only.*
6. *Perform quantifier-free cut-elimination on* ψ *to obtain a proof* φ *of* $\mathrm{sk}(S)$ *containing no non-atomic cuts.*

Let us remark here that in step 6, any method for cut-elimination for quantifier-free cuts can be used (e.g. reductive methods, "zero-th order" CERES). Furthermore, considering that the instantiations of quantifiers are the core information in a proof, one can even leave out this step as the instantiations in φ and ψ coincide.

Theorem 1. *Let* π *be a* **QFC***-proof of* S*. Then the* CERES^2 *method transforms* π *into an* $\mathbf{LKDe^2}$*-proof* φ *of* $\mathrm{sk}(S)$ *such that* φ *is in atomic-cut normal form.*

Proof. Using Proposition 1, we convert π to π_{sk}. By Lemma 3, there exists an R-refutation γ of $\mathrm{CL}(\pi_{sk})$. By Lemma 2, from γ we can construct an $\mathbf{LKDe^2}$-refutation γ' of $\mathrm{CL}(\pi_{sk})$. Every initial sequent of γ' is either a sequent $A \vdash A$, an axiom, or an instance C^* of some $C \in \mathrm{CL}(\pi_{sk})$ under a substitution σ. Let $C \equiv \Pi \vdash \Lambda$ and $\mathrm{sk}(S) \equiv \Gamma \vdash \Delta$, then by Definition 10 we have a cut-free **QFC**-proof ψ_C of $\Gamma, \Pi \vdash \Lambda, \Delta$. Let $C^* \equiv \Pi^* \vdash \Lambda^*$, then by Lemma 1, we can construct $\mathbf{LKDe^2}$-proofs ψ_{C^*} of $\Gamma, \Pi^* \vdash \Lambda^*, \Delta$ that contain quantifier-free cuts only. By plugging these proofs onto the leaves of γ' and adding contractions at the end, we obtain an $\mathbf{LKDe^2}$-proof of $\Gamma \vdash \Delta$ containing quantifier-free cuts only. By applying cut-elimination to this proof, we obtain the desired proof φ.

4.1 Extending $\mathbf{CERES^2}$

This work defines a method for cut-elimination for **QFC**-proofs. A natural question is then, whether the method can be extended to stronger comprehension. In the previous section, it was stated that skolemization is an important technical tool in the context of the method, as it removes strong quantifier introduction rules and because of this allows the definition of proof projections without causing violations of eigenvariable conditions.

When considering comprehension involving quantifiers, proof skolemization has to be modified to achieve the same effect: it is not enough to skolemize the end-sequent, as the active formulas of strong quantifier rules may be ancestors of formulas removed by weak second-order quantifier rules and therefore, the corresponding strong quantifiers will not be present in the end-sequent.

A tempting idea is, then, to simply skolemize the formulas that disappear into weak second-order quantifier rules. This approach is investigated in [6] and it turns out that weak quantifier rules cannot be skolemized within $\mathbf{LKDe^2}$ in most cases; the class where this is possible is only slightly larger than **QFC** and looks rather unnatural. So either we have to extend proof skolemization to more involved proof transformations or new techniques for dealing with projections containing strong quantifier rules have to be developed. A promising approach

is to use strong quantifier rules which introduce a quantifier not from a free variable but from a Skolem term as in [13].

5 CERES^2 Example

We will now apply the CERES^2 method to a **QFC**-proof φ. The proof under consideration is a proof of the theorem $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ by the least number principle. As axioms the proof uses elementary axioms of arithmetic such as associativity and commutativity of $+$ and $*$, axioms for the neutral elements 0 and 1, and distributivity. The following axioms represent the recursive definition of the series:

$$\vdash \Sigma(n+1) = \Sigma(n) + (n+1) \; ; \vdash \Sigma(0) = 0$$

We write 2 for $1+1$. In the proof, $\star$ denotes the ancestors of a cut, double lines indicate applications of propositional rules, and structural rules except cut are omitted.

$\varphi :=$

$$\dfrac{\begin{array}{cc} \varphi_1 & \varphi_2 \\ \vdots & \vdots \\ LNP \vdash IND^\star & IND^\star \vdash (\forall n) 2 * \Sigma(n) = n * (n+1) \end{array}}{LNP \vdash (\forall n) 2 * \Sigma(n) = n * (n+1)}\, cut$$

where

$$LNP \equiv (\forall Y)((\exists z) z \in Y \to 0 \in Y \vee (\exists z)(z \notin Y \wedge z+1 \in Y))$$
$$IND \equiv (\forall X)(0 \in X \wedge (\forall y)(y \in X \to y+1 \in X) \to (\forall y) y \in X)$$

This proof uses the fact that the least number principle implies induction as a lemma; the use of this lemma will be removed by application of the CERES^2 method, yielding a new proof that shows that the least number principle implies the theorem, without the use of induction.

The proof φ_1 specified below is exactly the proof of this lemma, and it is a formalization of the following argument: Assume the least number principle, and assume that for an arbitrary set $\mathcal{X}$, $0 \in \mathcal{X}$ and if $y \in \mathcal{X}$, then $y+1 \in \mathcal{X}$, and assume for contradiction that $\mathcal{X} \neq \mathbb{N}$. Then the set $\bar{\mathcal{X}} = \{x \mid x \notin \mathcal{X}\}$ (or $\lambda x.x \notin X$ in the lambda notation) is not empty, so by the least number principle either

1. $0 \in \bar{\mathcal{X}}$. But $0 \in \mathcal{X}$ by assumption, so $0 \notin \bar{\mathcal{X}}$.
2. There is a z s.t. $z \notin \bar{\mathcal{X}}$ and $z+1 \in \bar{\mathcal{X}}$. But then $z \in \mathcal{X}$ and by assumption $z+1 \in \mathcal{X}$, so $z+1 \notin \bar{\mathcal{X}}$.

So φ_1 is

$$\dfrac{\dfrac{\dfrac{\dfrac{y_0 \in X_0 \vdash y_0 \in X_0^\star}{\vdash (\forall y) y \in X_0^\star, (\exists z) z \notin X_0}\, \forall : r, \exists : r \qquad \varphi_1^1}{0 \in X_0^\star, (\forall y)(y \in X_0 \to y+1 \in X_0)^\star, LNP_\sigma \vdash (\forall y) y \in X_0^\star}\, \to : l}{LNP \vdash 0 \in X_0 \wedge (\forall y)(y \in X_0 \to y+1 \in X_0) \to (\forall y) y \in X_0^\star}\, \forall^2 : l\ \lambda x.x \notin X_0}{LNP \vdash IND^\star}\, \forall^2 : r$$

where

$$LNP_\sigma \equiv (\exists z) z \notin X_0 \to 0 \notin X_0 \vee (\exists z)(\neg z \notin X_0 \wedge z+1 \notin X_0)$$

The proof φ_1^1 is

$$\dfrac{\dfrac{0 \in X_0^\star \vdash 0 \in X_0}{0 \in X_0^\star, 0 \notin X_0 \vdash}\neg:l \qquad \dfrac{\dfrac{z_0 \in X_0 \vdash z_0 \in X_0^\star \qquad z_0+1 \in X_0^\star \vdash z_0+1 \in X_0}{z_0 \in X_0 \to z_0+1 \in X_0^\star, \neg z_0 \notin X_0 \wedge z_0+1 \notin X_0 \vdash}}{(\forall y)(y \in X_0 \to y+1 \in X_0)^\star, (\exists z)(\neg z \notin X_0 \wedge z+1 \notin X_0) \vdash}\exists,\forall:l}{0 \in X_0^\star, (\forall y)(y \in X_0 \to y+1 \in X_0)^\star, 0 \notin X_0 \vee (\exists z)(\neg z \notin X_0 \wedge z+1 \notin X_0) \vdash}\vee:l$$

This completes the left hand side of the cut, showing that the least number principle implies induction. The right hand side of the cut is a formalization of the following induction proof of $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$: The induction base is trivial. For the induction step we want to show

$$\sum_{i=0}^{n+1} i = n+1+\sum_{i=0}^{n} i = \frac{(n+1)((n+1)+1)}{2}$$

By the induction hypothesis this reduces to showing

$$n+1+\frac{n(n+1)}{2} = \frac{(n+1)((n+1)+1)}{2}$$

which clearly holds.

The formalization of this argument is the proof φ_2:

$$\dfrac{\varphi_2^1 \qquad \dfrac{\dfrac{2*\Sigma(n_0) = n_0*(n_0+1)^\star \vdash 2*\Sigma(n_0) = n_0*(n_0+1)}{(\forall x)2*\Sigma(x) = x*(x+1)^\star \vdash (\forall n)2*\Sigma(n) = n*(n+1)}\forall:r,\forall:l}{IND_\sigma^\star \vdash (\forall n)2*\Sigma(n) = n*(n+1)}\to:l}{IND^\star \vdash (\forall n)2*\Sigma(n) = n*(n+1)}\forall^2:l\ \lambda x.2*\Sigma(x) = x*(x+1)$$

where

$$IND_\sigma \equiv 2*\Sigma(0) = 0*(0+1) \wedge (\forall x)(2*\Sigma(x) = x*(x+1) \to 2*\Sigma(x+1) = (x+1)*((x+1)+1)) \to (\forall x)2*\Sigma(x) = x*(x+1)$$

We continue with φ_2^1 — from this point on, we will omit $\star$ as all formula occurrences in the following proofs are cut ancestors:

$$\dfrac{\dfrac{\vdash \Sigma(0) = 0 \qquad \dfrac{\vdash 2*0 = 0 \qquad \vdash 0 = 0*(0+1)}{\vdash 2*0 = 0*(0+1)}=:r_2}{\vdash 2*\Sigma(0) = 0*(0+1)}=:r_2 \qquad \varphi_2^2}{\vdash 2*\Sigma(0) = 0*(0+1) \wedge (\forall x)(2*\Sigma(x) = x*(x+1) \to 2*\Sigma(x+1) = (x+1)*((x+1)+1))}\wedge$$

Note that the left branch of φ_2^1 proves the induction base. The proof φ_2^2 will in turn show the induction step:

$$\dfrac{\dfrac{\vdash \Sigma(x_0+1) = \Sigma(x_0)+(x_0+1) \qquad \varphi_1^=}{2*\Sigma(x_0) = x_0*(x_0+1) \vdash 2*\Sigma(x_0+1) = (x_0+1)*((x_0+1)+1)}=:r_2}{\vdash (\forall x)(2*\Sigma(x) = x*(x+1) \to 2*\Sigma(x+1) = (x+1)*((x+1)+1))}\forall:r$$

where $\varphi_1^=$ is a proof of

$$2 * \Sigma(x_0) = x_0 * (x_0 + 1) \vdash 2 * (\Sigma(x_0) + (x_0 + 1)) = (x_0 + 1) * ((x_0 + 1) + 1)$$

using purely equational reasoning. This completes the proof φ.

Skolemization of φ (for details on proof skolemization, refer to the proof of Proposition 1 in [6]) yields a proof φ_{sk} of the sequent

$$(\forall Y)((\exists z) z \in Y \to 0 \in Y \vee (f(Y) \notin Y \wedge f(Y) + 1 \in Y)) \vdash 2 * \Sigma(s) = s * (s + 1)$$

where f, s are the Skolem symbols. In the proof, the Skolem term $f(\lambda x.x \notin X_0)$ replaces the eigenvariable z_0 and the Skolem term s replaces the eigenvariable n_0.

Remark 1. In all models of arithmetic and the left hand side of the sequent, a suitable interpretation of f will be a function $\gamma : P(\mathbb{N}) \mapsto \mathbb{N}$ such that for all $S \in P(\mathbb{N})$ with $S \neq \emptyset, 0 \notin S$, we have $\gamma(S) = \min(S) - 1$. This is an example for the natural interpretation of Skolem symbols, which in practice is often possible.

The characteristic clause set $\mathrm{CL}(\varphi_{sk})$ can be written as:

$$\mathrm{CL}(\varphi_{sk}) = \mathrm{CL}(\varphi_{sk}^1) \cup \mathrm{CL}(\varphi_{sk}^2)$$

$$\begin{aligned}\mathrm{CL}(\varphi_{sk}^1) = (\{0 \in X_0 \vdash\} \times \{\vdash f(\lambda x.x \notin X_0) \in X_0; f(\lambda x.x \notin X_0) + 1 \in X_0 \vdash\}) \\ \times \{\vdash y_0 \in X_0\}\end{aligned}$$

$$\begin{aligned}\mathrm{CL}(\varphi_{sk}^2) = \{2 * \Sigma(s) = s * (s + 1) \vdash\} \cup \{\vdash \Sigma(x_0 + 1) = \Sigma(x_0) + (x_0 + 1)\} \\ \cup \{2 * \Sigma(x_0) = x_0 * (x_0 + 1) \vdash 2 * \Sigma(x_0) = x_0 * (x_0 + 1)\} \\ \cup \{\vdash \Sigma(0) = 0\} \cup PAX_S\end{aligned}$$

where PAX_S is the set of axioms of arithmetic that are used in the proof φ_2^1. Modulo subsumption and tautology deletion, the characteristic clause set is:

$$\begin{aligned}CL(\varphi_{sk}) = \{\ & 0 \in X_0 \vdash f(\lambda x.x \notin X_0) \in X_0, y_0 \in X_0; && (I1) \\ & 0 \in X_0, f(\lambda x.x \notin X_0) + 1 \in X_0 \vdash y_0 \in X_0; && (I2) \\ & 2 * \Sigma(s) = s * (s + 1) \vdash; && (T1) \\ & \vdash \Sigma(x_0 + 1) = \Sigma(x_0) + (x_0 + 1); && (S1) \\ & \vdash \Sigma(0) = 0\} && (S2) \\ & \cup PAX'_S\end{aligned}$$

where PAX'_S is PAX_S after subsumption and tautology deletion.

5.1 Refutation of the Characteristic Clause Set

We now define a resolution refutation of the characteristic clause set $CL(\varphi_{sk})$, using the resolution calculus from Section 3.

The clauses $(I1)$ and $(I2)$ correspond to the induction axiom, while the clause $(T1)$ is the negated theorem. For the refutation we will need the following instances of the induction clauses produced from the substitution

$\sigma = \langle \{y_0 \leftarrow s\}, \{X_0 \leftarrow \lambda x.2 * \Sigma(x) = x * (x+1)\}\rangle$:

$(I1')\ 2 * \Sigma(0) = 0 * (0+1)$
$\quad \vdash 2 * \Sigma(f(T)) = f(T) * (f(T)+1), 2 * \Sigma(s) = s * (s+1)$

$(I2')\ 2 * \Sigma(0) = 0 * (0+1), 2 * \Sigma(f(T)+1) = (f(T)+1) * ((f(T)+1)+1)$
$\quad \vdash 2 * \Sigma(s) = s * (s+1)$

where $T \equiv \lambda x.\neg 2 * \Sigma(x) = x * (x+1)$. We start by deriving the induction base using resolution, for this we need the clauses

$$(A1) \vdash 2 * 0 = 0\ ;\ (A2) \vdash 0 = 0 * (0+1)$$

Note that $(A1), (A2) \in PAX_S$. We now use paramodulation from $(S2)$ into $(A1)$ to derive

$$(IB1) \vdash 2 * \Sigma(0) = 0$$

Paramodulation from $(IB1)$ into $(A2)$ then yields

$$(IB) \vdash 2 * \Sigma(0) = 0 * (0+1)$$

We now resolve both $(I1')$ and $(I2')$ first with (IB) and then with $(T1)$ to obtain

$$(IH) \vdash 2 * \Sigma(f(T)) = f(T) * (f(T)+1)$$
$$(IG)\ 2 * \Sigma(f(T)+1) = (f(T)+1) * ((f(T)+1)+1) \vdash$$

Note that (IH) corresponds to the induction hypothesis in the original proof, while (IG) is the negation of what was proved in the induction step. Towards a contradiction, we paramodulate (IG) with an instance of the second part of the definition of the series, $(S1)$, and get

$$(C1)\ 2 * (\Sigma(f(T)) + (f(T)+1)) = (f(T)+1) * ((f(T)+1)+1) \vdash$$

From clauses from PAX_S, it is easy to derive (using paramodulation exclusively) the clause

$$(C2) \vdash 2 * (\Sigma(x_0) + (x_0+1)) = 2 * \Sigma(x_0) + 2 * (x_0+1)$$

Paramodulation from an instance of $(C2)$ into $(C1)$ yields

$$(C3)\ 2 * \Sigma(f(T)) + 2 * (f(T)+1) = (f(T)+1) * ((f(T)+1)+1) \vdash$$

We can now use paramodulation to obtain from $(C3)$ and (IH) the clause

$$(C4)\ (f(T) * (f(T)+1)) + 2 * (f(T)+1) = (f(T)+1) * ((f(T)+1)+1) \vdash$$

which is a wrong arithmetical statement. From clauses in PAX_S it is now easy to derive the dual clause (modulo substitution)

$$(C5) \vdash x_0 * (x_0+1) + 2 * (x_0+1) = (x_0+1) * ((x_0+1)+1)$$

We can now resolve $(C4)$ with an instance of $(C5)$ to obtain the empty sequent and complete the refutation. Note that although the clauses used in the refutation correspond to the induction axiom, the proof constructed from the refutation will be a proof by the least number principle. This will become clear in the next section.

5.2 Interpretation of the ACNF

In this section we will indicate the construction of the ACNF from the refutation of $CL(\varphi_{sk})$ produced in the previous section. We will not give the full ACNF in this section, as it is too large to display comfortably, but we will discuss its key features.

The key information of a cut-free proof lies in the instantiations of the quantifiers (the other information just pertains to propositional reasoning and structural manipulation of sequents), so we will investigate a projection that contains such instantiations, namely the projection $\varphi[(I1)]$ to the clause

$$(I1) \equiv 0 \in X_0 \vdash f(S) \in X_0, y_0 \in X_0$$

where $S \equiv \lambda x.x \notin X_0$:

$$\dfrac{\dfrac{\dfrac{\dfrac{y_0 \in X_0 \vdash y_0 \in X_0}{\vdash y_0 \notin X_0, y_0 \in X_0}\neg : r}{\vdash (\exists z) z \notin X_0, y_0 \in X_0}\exists : r \qquad \psi}{0 \in X_0, (\exists z) z \notin X_0 \to (0 \notin X_0 \vee (\neg f(S) \notin X_0 \wedge f(S)+1 \notin X_0)) \vdash \Delta}\to : l}{0 \in X_0, (\forall Y)((\exists z) z \in Y \to (0 \in Y \vee (f(Y) \notin Y \wedge f(Y)+1 \in Y))) \vdash \Delta}\forall^2 : l\ \lambda x.x \notin X_0$$

where $\Delta \equiv f(S) \in X_0, y_0 \in X_0$ and ψ only consists of propositional inferences from tautological initial sequents. In the refutation, the instance of $(I1)$ under the substitution $\sigma = \langle \{y_0 \leftarrow s\}, \{X_0 \leftarrow \lambda x.2 * \Sigma(x) = x * (x+1)\}\rangle$ is used, therefore the projection used in the construction of the ACNF is $\varphi[(I1)]\sigma$ (cf. Lemma 1). This yields a projection with a rule application $\forall^2 : l\ \lambda x.\neg(2 * \Sigma(x) = x * (x+1))$. This use of comprehension is the key point in the argument of the ACNF: while the proof by induction showed that the formula holds for all n (or in other words, all n are in the set X), the proof by the least number property shows that the negation of the formula holds for no n (or that $\bar{X}$ is empty).

It is interesting to note that no matter what theorem is proved by induction in the input proof, the proof of $LNP \vdash IND$ remains the same and therefore also $(I1)$ and $\varphi[(I1)]$ remain unchanged. So as long as the clause $(I1)$ is used in the resolution refutation, the resulting ACNF will contain the above argument where only the definition of the set X differs.

6 Future Work and Acknowledgment

There is still much to be done: We are working on extending CERES^2 to larger classes of proofs and investigating the use of existing higher-order resolution calculi (see e.g. [15]) with CERES^2. For semi-automated application of the method, it will be necessary to replace the unrestricted substitution of our resolution calculus by unification (see e.g. [16]). Also, the existing ANSI C++ implementation of CERES is being extended to CERES^2. This will allow practical application of the method to larger and more interesting proofs.

Finally, we would like to thank the anonymous referees for their helpful comments and suggestions for improvement of this paper.

References

1. Kohlenbach, U.: Effective bounds from ineffective proofs in analysis: an application of functional interpretation and majorization. Journal of Symbolic Logic 57(4), 1239–1273 (1992)
2. Baaz, M., Leitsch, A.: Towards a clausal analysis of cut-elimination. Journal of Symbolic Computation 41, 381–410 (2006)
3. Baaz, M., Leitsch, A.: Cut-elimination and Redundancy-elimination by Resolution. Journal of Symbolic Computation 29(2), 149–176 (2000)
4. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: Ceres: An Analysis of Fürstenberg's Proof of the Infinity of Primes. Theoretical Computer Science 403, 160–175 (2008)
5. Hetzl, S.: Characteristic Clause Sets and Proof Transformations. PhD thesis, Vienna University of Technology (2007)
6. Hetzl, S., Leitsch, A., Weller, D., Woltzenlogel Paleo, B.: CERES in second-order logic. Technical report, Vienna University of Technology (2008), `http://www.logic.at/ceres/downloads/docs/report_ceres2.pdf`
7. Church, A.: A formulation of the simple theory of types. JSL 5(2), 56–68 (1940)
8. Boolos, G.S., Burgess, J.P., Jeffrey, R.C.: Computability and Logic, 4th edn. Cambridge University Press, Cambridge (2002)
9. Takeuti, G.: Proof Theory. Studies in Logic and the Foundations of Mathematics, vol. 81. North-Holland Publishing Co., Amsterdam (1975)
10. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: Proof transformation by CERES. In: Borwein, J.M., Farmer, W.M. (eds.) MKM 2006. LNCS (LNAI), vol. 4108, pp. 82–93. Springer, Heidelberg (2006)
11. Andrews, P.B.: Resolution in Type Theory. Journal of Symbolic Logic 36(3), 414–432 (1971)
12. Baaz, M., Leitsch, A.: Cut normal forms and proof complexity. Annals of Pure and Applied Logic 97, 127–177 (1999)
13. Miller, D.A.: A compact representation of proofs. Studia Logica 46(4), 347–370 (1987)
14. Danos, V., Joinet, J.B., Schellinx, H.: A New Deconstructive Logic: Linear Logic. Journal of Symbolic Logic 62(3), 755–807 (1997)
15. Benzmüller, C.: Comparing approaches to resolution based higher-order theorem proving. Synthese 133(1-2), 203–335 (2002)
16. Dowek, G.: Higher-order unification and matching. In: Handbook of automated reasoning, pp. 1009–1062. Elsevier Science Publishers B.V., Amsterdam (2001)

Hypersequent Systems for the Admissible Rules of Modal and Intermediate Logics

Rosalie Iemhoff[1] and George Metcalfe[2]

[1] Department of Philosophy, Utrecht University
Bestuursgebouw, Heidelberglaan 6-8, 3584 CS Utrecht, The Netherlands
Rosalie.Iemhoff@phil.uu.nl
[2] Department of Mathematics, Vanderbilt University
1326 Stevenson Center, Nashville TN 37240, USA
george.metcalfe@vanderbilt.edu

Abstract. The admissible rules of a logic are those rules under which the set of theorems of the logic is closed. In a previous paper by the authors, formal systems for deriving the admissible rules of Intuitionistic Logic and a class of modal logics were defined in a proof-theoretic framework where the basic objects of the systems are sequent rules. Here, the framework is extended to cover derivability of the admissible rules of intermediate logics and a wider class of modal logics, in this case, by taking hypersequent rules as the basic objects.

1 Introduction

Investigations into logical systems typically focus on the derivability of formulas or other structures within the system. However, the admissibility of rules for the system may also play a key role. A rule is *admissible* for a logic (viewed as a consequence relation) if adding it to the logic produces no new theorems. Such a notion is of interest in Computer Science for (at least) two reasons. First, admissible rules show that the derivability of certain formulas implies the derivability of stronger formulas, in the sense that the latter derive the former but not vice versa, an example being the disjunction property, where the derivability of a disjunction implies that one of the disjuncts is derivable. Second, equational unification can be formulated in terms of admissible rules. A formula A is unifiable for a consistent logic L iff σA is a theorem of L for some substitution σ. But this is equivalent to the claim that the rule A/q is not admissible in L for any variable q not occurring in A. Finally, admissible rules are also interesting from an algebraic perspective: they correspond to quasi-equations holding in the free algebra with countably many generators (or the Lindenbaum algebra of the logic).

Classical Logic has no non-derivable admissible rules; that is, it is structurally complete. However, for non-classical logics, this is no longer the case, and it is an interesting and often quite challenging task to provide characterizations of admissibility for these logics. In the case of modal and intermediate logics, a wide range of results for admissible rules such as decidability and complexity have been obtained, in particular by Rybakov [13]. Axiomatic-style presentations have been provided for wide classes of intermediate logics by Iemhoff [7,8] (the case of Intuitionistic Logic was considered

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 230–245, 2009.

independently by Rozière [12]) and modal logics by Jeřábek [10], both making crucial use of Ghilardi's work on unification and projective approximations [4,5].

In [9], the current authors introduced a proof-theoretic framework for admissibility: analytic "Gentzen-style" proof systems for deriving the admissible rules of both Intuitionistic Logic and a class of "extensible" modal logics including $\mathsf{K4}$, $\mathsf{S4}$, and GL. The key idea of this approach is that just as calculi for derivability in these logics can often be presented using sequents, so the corresponding systems for admissibility can be presented using sequent rules as basic objects. Here, we extend this approach to both a class of intermediate logics, including De Morgan Logic KC and the bounded cardinality logics $\mathsf{BC}_1, \mathsf{BC}_2, \ldots$, and a wider class of "mono-extensible" modal logics, including logics such as $\mathsf{GL.3}$, $\mathsf{S4.2}$, etc. In this case, however, the natural home for derivability is not sequents, but the framework of *hypersequents* – intuitively, disjunctions of sequents – introduced by Avron in [1] and used to define calculi for families of both intermediate logics (see e.g. [3]) and fuzzy logics (see e.g. [11]). Hence, for admissibility in these logics, the basic objects of our systems will be hypersequent rules.

2 Admissible Rules

Let us assume for this paper that the logic L is treated as a consequence relation based on a propositional language with binary connectives $\wedge, \vee, \rightarrow$, a constant $\bot$, and sometimes also a modal connective $\Box$. Other connectives are then defined as:

$$\neg A =_{\text{def}} A \rightarrow \bot \quad \top =_{\text{def}} \neg\bot \quad A \leftrightarrow B =_{\text{def}} (A \rightarrow B) \wedge (B \rightarrow A) \quad \boxdot A =_{\text{def}} \Box A \wedge A$$

We denote (propositional) variables by $p, q, r, \ldots$, formulas by $A, B, C, \ldots$, and finite sets of formulas by $\Gamma, \Pi, \Sigma, \Delta, \Theta, \Psi$. Formulas $p \rightarrow q$ and $\Box p$ are called *variable implications* and *boxed variables*, respectively. We also write $\bigvee \Gamma$ and $\bigwedge \Gamma$ where $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$ for iterated disjunctions and conjunctions of formulas in a finite set Γ, and make use of the abbreviations:

$$\Box\Gamma =_{\text{def}} \{\Box A : A \in \Gamma\} \quad \boxdot \Gamma =_{\text{def}} \Gamma \cup \Box\Gamma \quad (\Gamma \equiv \Box\Gamma) =_{\text{def}} \{A \leftrightarrow \Box A : A \in \Gamma\}$$

Typically, logical rules are asymmetric, having many premises but just one conclusion. However, for admissibility, it is convenient to treat instead *generalized rules* of the form $\Gamma \rhd \Delta$, where both Γ and Δ are sets of formulas. Intuitively, such a rule is admissible for a logic L if whenever a substitution makes all the premises theorems of L, it also makes one of the conclusions a theorem. More precisely, an L-*unifier* for a formula A is a substitution σ such that $\vdash_{\mathsf{L}} \sigma A$. Then a generalized rule $\Gamma \rhd \Delta$ is L-*admissible*, written $\Gamma \mathrel{\sim_{\mathsf{L}}} \Delta$, if each L-unifier for all $A \in \Gamma$, is an L-unifier for some $B \in \Delta$.

Example 1. A nice example of an admissible generalized rule for Intuitionistic Logic is the *disjunction property*, formulated as $p \vee q \rhd p, q$. If $\vdash_{\mathsf{IPC}} \sigma(p) \vee \sigma(q)$, then either $\vdash_{\mathsf{IPC}} \sigma(p)$ or $\vdash_{\mathsf{IPC}} \sigma(q)$. However, this rule is not admissible for Classical Logic; e.g. for $\sigma(p) = p$ and $\sigma(q) = \neg p$, plainly $\vdash_{\mathsf{CPC}} p \vee \neg p$, but $\nvdash_{\mathsf{CPC}} p$ and $\nvdash_{\mathsf{CPC}} \neg p$.

Although admissibility and derivability do not coincide in general for non-classical logics, Ghilardi in [4,5] identified classes of "projective" formulas A where the relationship

"$A \vdash\!\sim_{\mathsf{L}} B$ iff $A \vdash_{\mathsf{L}} B$" holds for all formulas B. Let us make this precise. A formula A is L-*projective* for a logic L if there exists a substitution σ, called an L-*projective unifier* for A, such that $\vdash_{\mathsf{L}} \sigma A$ and $A \vdash_{\mathsf{L}} \sigma(p) \leftrightarrow p$ for all variables p.

Lemma 1. *Let* L *be an intermediate logic or a normal extension of* $\mathsf{K4}$*:*

(a) If A is L-projective, then $A \vdash\!\sim_{\mathsf{L}} \Delta$ iff $A \vdash_{\mathsf{L}} B$ for some $B \in \Delta$.
(b) If $A_1, \ldots, A_n$ are L-projective, then $\bigvee_{i=1}^n A_i \vdash\!\sim_{\mathsf{L}} B$ iff $\bigvee_{i=1}^n A_i \vdash_{\mathsf{L}} B$.
(c) If L' extends L (as a consequence relation) and $A_1, \ldots, A_n$ are L-projective formulas, then $\bigvee_{i=1}^n A_i \vdash\!\sim_{\mathsf{L}'} B$ iff $\bigvee_{i=1}^n A_i \vdash_{\mathsf{L}'} B$.
(d) If L' extends a normal modal logic L (as a consequence relation) and $A_1, \ldots, A_n$ are L-projective formulas, then $\bigvee_{i=1}^n \Box A_i \vdash\!\sim_{\mathsf{L}'} \Box B$ iff $\bigvee_{i=1}^n \Box A_i \vdash_{\mathsf{L}'} \Box B$.

Proof. (a) The right-to-left direction is immediate. For the other direction, suppose that $A \vdash\!\sim_{\mathsf{L}} \Delta$ where A is L-projective. Then there exists an L-projective unifier σ of A, such that $\vdash_{\mathsf{L}} \sigma B$ for some $B \in \Delta$. Also $A \vdash_{\mathsf{L}} \sigma B \to B$, so by modus ponens, $A \vdash_{\mathsf{L}} B$. (b) Again, the right-to-left direction is immediate. For the other direction, suppose that $\bigvee_{i=1}^n A_i \vdash\!\sim_{\mathsf{L}} B$. Also then $A_i \vdash\!\sim_{\mathsf{L}} B$ for $i = 1 \ldots n$. By (a), $A_i \vdash_{\mathsf{L}} B$ for $i = 1 \ldots n$ and hence $\bigvee_{i=1}^n A_i \vdash_{\mathsf{L}} B$. For (c), let L' be an extension of L and let $A_1, \ldots, A_n$ be L-projective formulas. Since L' extends L, we get that $A_1, \ldots, A_n$ are also L'-projective. The result then follows from (b). For (d), as for (c), we get that $A_1, \ldots, A_n$ are L'-projective. Suppose that $\bigvee_{i=1}^n \Box A_i \vdash\!\sim_{\mathsf{L}'} \Box B$. Then also $\Box A_i \vdash\!\sim_{\mathsf{L}'} \Box B$ for $i = 1 \ldots n$. Hence $A_i \vdash\!\sim_{\mathsf{L}'} B$ and by (a), $A_i \vdash_{\mathsf{L}'} B$ for $i = 1 \ldots n$. But then $\Box A_i \vdash_{\mathsf{L}'} \Box B$ for $i = 1 \ldots n$ and hence $\bigvee_{i=1}^n \Box A_i \vdash_{\mathsf{L}'} \Box B$. The other direction is almost immediate. □

3 Modal Logics

In [9], formal systems were defined for deriving the admissible rules of *extensible modal logics* by taking sequent rules as basic objects. Below, we recall this characterization and show that it can be extended to a wider class of logics that we call *mono-extensible* by taking our basic objects to be hypersequent rules.

3.1 Extensible and Mono-extensible Modal Logics

For a comprehensive account of modal logics, see e.g. [2]. Let us just recall that for any normal modal logic L, an L-frame is such that every model on that frame is a model of L, and an L-model is a model based on an L-frame. L has the finite model property FMP if every refutable formula is refutable on a finite L-frame. For a Kripke model K with accessibility relation R, the *root* of K is the cluster $\{k : \forall l \neq k(kRl)\}$, and K_k denotes the Kripke model K restricted to the domain $\{l : kRl \text{ or } k = l\}$. Two Kripke models K_1, K_2 are *variants* of one another if they have the same nodes and accessibility relation, and their forcing relations agree on all nodes except possibly the root.

Ghilardi [5] has given a characterization for projectivity for a wide range of modal logics (following here the terminology of [10]).

Theorem 1 (Ghilardi [5]). *A class of finite models $\mathcal{K}$ has the* modal extension property *if for any model K, whenever $K_k \in \mathcal{K}$ for all k not in the root of K, there is a variant*

of K in $\mathcal{K}$. For every normal extension L *of* $\mathsf{K4}$ *with the* FMP*, a formula is* L*-projective iff its class of* L*-models has the modal extension property.*

To get a handle on the modal extension property, we recall two useful constructions.

Definition 1. *For frames $F_1, \ldots, F_n$, $(\sum F_j)^i$ and $(\sum F_j)^r$ are obtained by adding, respectively, an irreflexive or a reflexive node beneath (connected to all nodes of) the disjoint sum of $F_1 \ldots F_n$. A normal extension* L *of* $\mathsf{K4}$ *with the* FMP *is* extensible *if for all finite* L*-frames $F_1, \ldots, F_n$:*

(i) $(\sum F_j)^i$ is an L*-frame unless* L *is reflexive;*
(ii) $(\sum F_j)^r$ is an L*-frame unless* L *is irreflexive.*

L *is* mono-extensible *if it satisfies the above for $n = 1$; that is, for each finite* L*-frame F: (i) F^i is an* L*-frame, unless* L *is reflexive; (ii) F^r is an* L*-frame, unless* L *is irreflexive.*

L *is* linear-extensible *if it is mono-extensible and linear; i.e. all rooted* L*-frames are linear (or* L *proves $\Box(\Box A \to B) \vee \Box(\Box B \to A)$).*

Every extensible logic L obeys the modal disjunction property: if $\vdash_{\mathsf{L}} \Box A \vee \Box B$, then $\vdash_{\mathsf{L}} \Box A$ or $\vdash_{\mathsf{L}} \Box B$. I.e. $\Box p \vee \Box q \rhd \Box p, \Box q$ is L-admissible. Significant examples of these logics include $\mathsf{K4}$, $\mathsf{S4}$, Grz, and GL. Linear-extensible logics (treated in [10]), which include the logics $\mathsf{S4.3}$, $\mathsf{K4.3}$, and $\mathsf{GL.3}$, and clearly do not satisfy the modal disjunction property, are the most obvious examples of mono-extensible but not extensible logics. Other interesting examples include the logics $\mathsf{S4.2}$, $\mathsf{K4.2}$, and $\mathsf{GL.2}$ (also discussed in [10]) which are mono-extensible but neither extensible nor linear-extensible.

3.2 Sequent Systems for Extensible Logics

Gentzen systems for derivability in many non-classical logics, in particular core modal logics, can be obtained in the framework of sequents. Since order and multiplicity of formulas is unimportant in the context of modal logics, we define a *sequent* S here as an ordered pair of finite sets of formulas, written $\Gamma \Rightarrow \Delta$. Such a sequent is said to be L-*derivable*, written $\vdash_{\mathsf{L}} S$, iff $\vdash_{\mathsf{L}} I(S)$ where $I(\Gamma \Rightarrow \Delta) =_{\text{def}} \bigwedge \Gamma \to \bigvee \Delta$.

To obtain Gentzen-style proof systems for admissibility in extensible modal logics, it is convenient again to use sequents, but this time at the level of rules. A *generalized sequent rule* (gs-rule for short) $\mathtt{R}$ is an ordered pair of finite sets of sequents, written:

$$\{\Gamma_i \Rightarrow \Delta_i\}_{i=1}^n \rhd \{\Pi_j \Rightarrow \Sigma_j\}_{j=1}^m$$

- $\mathtt{R}$ is L-*admissible*, written $\sim_{\mathsf{L}} \mathtt{R}$, iff $\{I(\Gamma_i \Rightarrow \Delta_i)\}_{i=1}^n \sim_{\mathsf{L}} \{I(\Pi_j \Rightarrow \Sigma_j)\}_{j=1}^m$.
- $\mathtt{R}$ is L-*derivable*, written $\vdash_{\mathsf{L}} \mathtt{R}$, iff $\bigwedge_{i=1}^n I(\Gamma_i \Rightarrow \Delta_i) \vdash_{\mathsf{L}} \bigvee_{j=1}^m I(\Pi_j \Rightarrow \Sigma_j)$.

Note that $A_1, \ldots, A_n \sim_{\mathsf{L}} B_1, \ldots, B_m$ iff $\sim_{\mathsf{L}} (\Rightarrow A_1), \ldots, (\Rightarrow A_n) \rhd (\Rightarrow B_1), \ldots, (\Rightarrow B_m)$. Hence a proof system for the admissibility of gs-rules is also a proof system for the admissibility of generalized rules, and of course, rules in the usual sense.

Rules (now at the next level up) for gs-rules are sets of rule instances, each consisting of a set of premises $\mathtt{R}_1, \ldots, \mathtt{R}_n$ and a conclusion $\mathtt{R}$; instances with no premises being called *initial gs-rules*. They are defined here schematically, using p, q to stand

Initial GS-Rules and Structural Rules

$$\frac{}{\mathcal{G} \rhd (\Gamma, A \Rightarrow A, \Delta), \mathcal{H}}\ (\text{ID}) \qquad \frac{\mathcal{G} \rhd \mathcal{H}}{\mathcal{G}, S \rhd \mathcal{H}}\ (\text{W})\rhd \qquad \frac{\mathcal{G} \rhd \mathcal{H}}{\mathcal{G} \rhd S, \mathcal{H}}\ \rhd(\text{W})$$

Anti-Cut and Projection Rules

$$\frac{\mathcal{G}, (\Gamma, A \Rightarrow \Delta), (\Pi \Rightarrow A, \Sigma), (\Gamma, \Pi \Rightarrow \Sigma, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \Rightarrow \Delta), (\Pi \Rightarrow A, \Sigma) \rhd \mathcal{H}}\ (\text{AC}) \qquad \frac{\mathcal{G}, S \rhd (\Gamma, \boxdot I(S) \Rightarrow \Delta), \mathcal{H}}{\mathcal{G}, S \rhd \mathcal{H}}\ (\text{PJ})$$

where $(\Gamma \Rightarrow \Delta) \in \mathcal{H} \cup \{\Rightarrow\}$

Right Logical Rules

$$\frac{}{\mathcal{G} \rhd (\Gamma, \bot \Rightarrow \Delta), \mathcal{H}}\ \rhd(\bot\Rightarrow) \qquad \frac{\mathcal{G} \rhd (\Gamma \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma \Rightarrow \bot, \Delta), \mathcal{H}}\ \rhd(\Rightarrow\bot)$$

$$\frac{\mathcal{G} \rhd (\Gamma \Rightarrow A, \Delta), \mathcal{H} \quad \mathcal{G} \rhd (\Gamma \Rightarrow B, \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma \Rightarrow A \wedge B, \Delta), \mathcal{H}}\ \rhd(\Rightarrow\wedge) \qquad \frac{\mathcal{G} \rhd (\Gamma, A, B \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma, A \wedge B \Rightarrow \Delta), \mathcal{H}}\ \rhd(\wedge\Rightarrow)$$

$$\frac{\mathcal{G} \rhd (\Gamma, A \Rightarrow \Delta), \mathcal{H} \quad \mathcal{G} \rhd (\Gamma, B \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma, A \vee B \Rightarrow \Delta), \mathcal{H}}\ \rhd(\vee\Rightarrow) \qquad \frac{\mathcal{G} \rhd (\Gamma \Rightarrow A, B, \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma \Rightarrow A \vee B, \Delta), \mathcal{H}}\ \rhd(\Rightarrow\vee)$$

$$\frac{\mathcal{G} \rhd (\Gamma \Rightarrow A, \Delta), \mathcal{H} \quad \mathcal{G} \rhd (\Gamma, B \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma, A \to B \Rightarrow \Delta), \mathcal{H}}\ \rhd(\to\Rightarrow) \qquad \frac{\mathcal{G} \rhd (\Gamma, A \Rightarrow B, \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma \Rightarrow A \to B, \Delta), \mathcal{H}}\ \rhd(\Rightarrow\to)$$

Left Logical Rules

$$\frac{\mathcal{G} \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, \bot \Rightarrow \Delta) \rhd \mathcal{H}}\ (\bot\Rightarrow)\rhd \qquad \frac{\mathcal{G}, (\Gamma \Rightarrow \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow \bot, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\bot)\rhd$$

$$\frac{\mathcal{G}, (\Gamma, A, B \Rightarrow \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \wedge B \Rightarrow \Delta) \rhd \mathcal{H}}\ (\wedge\Rightarrow)\rhd \qquad \frac{\mathcal{G}, (\Gamma \Rightarrow A, \Delta), (\Gamma \Rightarrow B, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow A \wedge B, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\wedge)\rhd$$

$$\frac{\mathcal{G}, (\Gamma \Rightarrow A, B, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow A \vee B, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\vee)\rhd \qquad \frac{\mathcal{G}, (\Gamma, A \Rightarrow \Delta), (\Gamma, B \Rightarrow \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \vee B \Rightarrow \Delta) \rhd \mathcal{H}}\ (\vee\Rightarrow)\rhd$$

$$\frac{\mathcal{G}, (\Gamma, B \Rightarrow \Delta), (\Gamma \Rightarrow A, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \to B \Rightarrow \Delta) \rhd \mathcal{H}}\ (\to\Rightarrow)\rhd \qquad \frac{\mathcal{G}, (\Gamma, A \Rightarrow B, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow A \to B, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\to)\rhd$$

$$\frac{\mathcal{G}, (\Gamma, \Box p \Rightarrow \Delta), (A \Rightarrow p) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, \Box A \Rightarrow \Delta) \rhd \mathcal{H}}\ (\Box\Rightarrow)\rhd \qquad \frac{\mathcal{G}, (\Gamma \Rightarrow \Box p, \Delta), (p \Rightarrow A) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow \Box A, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\Box)\rhd$$

where p does not occur in $\mathcal{G}, \mathcal{H}, \Gamma, \Delta, A$ in $(\Box\Rightarrow)\rhd$ and $(\Rightarrow\Box)\rhd$.

Fig. 1. Core Modal Rules

$$\frac{[\mathcal{G}, (\Box\Gamma \Rightarrow \Box\Delta), (\boxdot\Gamma \Rightarrow A) \rhd \mathcal{H}]_{A\in\Delta}}{\mathcal{G}, (\Box\Gamma \Rightarrow \Box\Delta) \rhd \mathcal{H}}\ (\text{V}^i)$$

$$\frac{[\mathcal{G}, (\Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta), (\boxdot\Gamma \Rightarrow A) \rhd \mathcal{H}]_{A\in\Delta}}{\mathcal{G}, (\Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta) \rhd \mathcal{H}}\ (\text{V}^r)$$

$$\frac{\mathcal{G}, (\Gamma, \Theta \Rightarrow \Delta), (\Pi \Rightarrow \Psi, \Sigma), (\Gamma, \Pi, A \leftrightarrow \Box A \Rightarrow \Sigma, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, \Theta \Rightarrow \Delta), (\Pi \Rightarrow \Psi, \Sigma) \rhd \mathcal{H}}\ (\text{AC}_\Box)$$

where $\Theta \cup \Psi \subseteq \{A, \Box A\}$ and $\Theta, \Psi \neq \emptyset$.

Fig. 2. Additional Extensible Modal Rules

for variables, A, B for formulas, $\Gamma, \Pi, \Sigma, \Delta, \Theta, \Psi$ for sets of formulas, S for sequents, and $\mathcal{G}, \mathcal{H}$ for sets of sequents. We call all sequents not in $\mathcal{G}$ or $\mathcal{H}$ for instances of such rules, *principal sequents*. Such rules are L-*sound* if whenever $\vdash_{\mathsf{L}} \mathtt{R}_i$ for $i = 1 \ldots n$, then $\vdash_{\mathsf{L}} \mathtt{R}$, and L-*invertible*, if whenever $\vdash_{\mathsf{L}} \mathtt{R}$, then $\vdash_{\mathsf{L}} \mathtt{R}_i$ for $i = 1 \ldots n$. Calculi for extensible modal logics are defined in this framework as follows:

Theorem 2 ([9]). *For an extensible modal logic* L *and gs-rule calculus* GAML *where:*

(1) GAML *extends the core modal rules of Figure 1.*
(2) If L *is not reflexive, then* (V^i) *is a rule of* GAML.
(3) If L *is not irreflexive, then* (V^r) *and* $(\mathrm{AC}_\Box)$ *are rules of* GAML.
(4) If $\vdash_{\mathsf{L}} S$, *then* $\vdash_{\mathsf{GAML}} \triangleright S$.
(5) If $\vdash_{\mathsf{GAML}} \mathtt{R}$, *then* $\vdash_{\mathsf{L}} \mathtt{R}$.

$\vdash_{\mathsf{L}} \mathtt{R}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any gs-rule* $\mathtt{R}$.

Note that the right logical rules of Figure 1 are just usual rules for modal logics embedded into the gs-rule framework, and that the non-modal left logical rules are obtained from these rules by replacing the conclusion sequent with the premise sequents (including $\triangleright(\bot \Rightarrow)$, an instance of $(\mathrm{W})\triangleright$ but included here for uniformity). The (non-invertible) modal rules, $(\Box\Rightarrow)\triangleright$ and $(\Rightarrow\Box)\triangleright$, decompose modal formulas on the left by replacing the formula A in $\Box A$ by a new variable p, soundness following from the fact that any substitution for the conclusion can be extended (since p does not occur there) by subsituting A for p. The "projection rule" (PJ) allows sequents on the left to be used as modal implications on the right, corresponding to the fact that derivability implies admissibility, while the "anti-cut" rule (AC) corresponds directly to the fact that the usual cut rule is admissible in the logic. The more complicated "Visser rules" (V^i) and (V^r) reflect the existence of non-derivable admissible rules for irreflexive and reflexive logics.

Example 2. In particular, we can obtain calculi for $\mathsf{K4}$, GL, and $\mathsf{S4}$ by adding (from left to right) the first rule for $\mathsf{K4}$, the second rule for GL, and the first and the third for $\mathsf{S4}$:

$$\frac{\mathcal{G} \rhd (\boxdot\Gamma \Rightarrow A), \mathcal{H}}{\mathcal{G} \rhd (\Box\Gamma, \Pi \Rightarrow \Box A, \Delta), \mathcal{H}} \qquad \frac{\mathcal{G} \rhd (\boxdot\Gamma, \Box A \Rightarrow A), \mathcal{H}}{\mathcal{G} \rhd (\Box\Gamma, \Pi \Rightarrow \Box A, \Delta), \mathcal{H}} \qquad \frac{\mathcal{G} \rhd (\boxdot\Gamma, \Pi \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Box\Gamma, \Pi \Rightarrow \Delta), \mathcal{H}}$$

to the core modal rules, with (V^i) for GL, (V^r) and $(\mathrm{AC}_\Box)$ for $\mathsf{S4}$, and all three for $\mathsf{K4}$.

3.3 Hypersequent Systems for Mono-extensible Logics

To deal with mono-extensible modal logics, we move beyond the sequent level. In particular, adapting slightly the usual definition (see e.g. Avron [1]), we define a *hypersequent* to be a finite non-empty set of sequents, written $S_1 \mid \ldots \mid S_n$, and let $\vdash_{\mathsf{L}} G$ iff $\vdash_{\mathsf{L}} I^{\Box}(G)$ where $I^{\Box}(G) = \bigvee_{i=1}^{n} \Box I(S_i)$. Hypersequent calculi are particularly useful for characterizing intermediate logics (see e.g. [3]) and logics characterized by linearly ordered structures [11]. An example of both is a calculus for the intermediate and fuzzy Gödel-Dummett Logic LC, defined by adding a single rule to a hypersequent version of Gentzen's calculus LJ for Intuitionistic Logic.

$$\frac{\mathcal{G}, G \rhd (\boxdot I^{\Box}(G) \Rightarrow I^{\Box}(H)), \mathcal{H}}{\mathcal{G}, G \rhd \mathcal{H}}\ (\text{PJ})^h \quad \text{where } H \in \mathcal{H} \cup \{\Rightarrow\}$$

$$\frac{\mathcal{G}, (G \mid \Box\Gamma \Rightarrow \Box\Delta), (G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\}) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Box\Gamma \Rightarrow \Box\Delta) \rhd \mathcal{H}}\ (\text{V}^i)^h$$

$$\frac{\mathcal{G}, (G \mid \Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta), (G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\}) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta) \rhd \mathcal{H}}\ (\text{V}^r)^h$$

$$\frac{\mathcal{G}, (G \mid \Gamma, A \Rightarrow \Delta), (H \mid \Pi \Rightarrow A, \Sigma)(G \mid H \mid \Gamma, \Pi \Rightarrow \Sigma, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Gamma, A \Rightarrow \Delta), (H \mid \Pi \Rightarrow A, \Sigma) \rhd \mathcal{H}}\ (\text{AC})^h$$

$$\frac{\mathcal{G}, (G \mid \Gamma, \Theta \Rightarrow \Delta), (H \mid \Pi \Rightarrow \Psi, \Sigma), (G \mid H \mid \Gamma, \Pi, A \leftrightarrow \Box A \Rightarrow \Sigma, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Gamma, \Theta \Rightarrow \Delta), (H \mid \Pi \Rightarrow \Psi, \Sigma) \rhd \mathcal{H}}\ (\text{AC}_{\Box})^h$$

$$\text{where } (\Theta \cup \Psi) \subseteq \{A, \Box A\} \text{ and } \Theta, \Psi \neq \emptyset$$

Fig. 3. Additional Mono-Extensible Modal Rules

Extending gs-rules to the hypersequent case, a *generalized hypersequent rule* (gh-rule for short) $\mathtt{R}$ is an ordered pair of sets of hypersequents, written:

$$G_1, \ldots, G_n \rhd H_1, \ldots, H_m$$

If $m \leq 1$, then $\mathtt{R}$ is called a *single-conclusion* gh-rule (sgh-rule for short).

- $\mathtt{R}$ is L-*admissible*, written $\mathsf{L} \mathtt{R}$, iff $\{I^{\Box}(G_i)\}_{i=1}^n \vdash_{\mathsf{L}} \{I^{\Box}(H_j)\}_{j=1}^m$.
- $\mathtt{R}$ is L-*derivable*, written $\vdash_{\mathsf{L}} \mathtt{R}$, iff $\bigwedge_{i=1}^n I^{\Box}(G_i) \vdash_{\mathsf{L}} \bigvee_{j=1}^m I^{\Box}(H_j)$.

A core set of rules for mono-extensible logics is obtained from the core modal rules by adding a context variable G in the premises and conclusion standing for an arbitrary context hypersequent; e.g. $(\Rightarrow\wedge)\rhd$ becomes:

$$\frac{\mathcal{G}, (G \mid \Gamma \Rightarrow A, \Delta), (G \mid \Gamma \Rightarrow B, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Gamma \Rightarrow A \wedge B, \Delta) \rhd \mathcal{H}}\ (\Rightarrow\wedge)\rhd^h$$

Hypersequent versions of the projection and anti-cut rules of Fig. 2 are given in Fig 3.

Definition 2. *The* gh-version $\mathtt{R}^h$ *of a rule schema* $\mathtt{R}$ *for gs-rules is obtained by replacing each principal sequent* S *in* $\mathtt{R}$ *by* $G \mid S$ *for a fixed hypersequent variable* G.

Our starting point for a calculus for a mono-extensible modal logic L is rules on the right of the $\rhd$ symbol that provide a sound and complete calculus for L-derivability. We then expand this calculus with hypersequent versions of the core modal rules, and versions of the appropriate Visser rules (V^i) and (V^r).

Definition 3. *A calculus* GAML *is* L*-fitting for a mono-extensible modal logic* L *if:*

(1) GAML *extends the gh-versions of the core modal rules.*
(2) If L *is not reflexive, then* $(\text{V}^i)^h$ *of Fig. 3 is a rule of* GAML.
(3) If L *is not irreflexive, then* $(\text{V}^r)^h$ *and* $(\text{AC}_{\Box})^h$ *of Fig. 3 are rules of* GAML.
(4) If $\vdash_{\mathsf{L}} G$, *then* $\vdash_{\mathsf{GAML}} \rhd G$.
(5) If $\vdash_{\mathsf{GAML}} \mathtt{R}$, *then* $\mathsf{L} \mathtt{R}$.

Example 3. For non-reflexive logics, we can use (v^i) as follows, noting that the top gh-rule is easily seen to be derivable using rules for $\mathsf{K4}$ on the right:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\triangleright\ \boxdot(\Box(\boxdot A \to B) \vee \Box(\boxdot A \Rightarrow C)) \Rightarrow \Box(\boxdot A \to (B \vee C))}}{(\boxdot A \Rightarrow B \mid \boxdot A \Rightarrow C)\ \triangleright\ \boxdot(\Box(\boxdot A \to B) \vee \Box(\boxdot A \Rightarrow C)) \Rightarrow \Box(\boxdot A \to (B \vee C))}\,(\mathrm{w})\triangleright}{(\boxdot A \Rightarrow B \mid \boxdot A \Rightarrow C)\ \triangleright\ (\boxdot A \Rightarrow B \vee C)}\,(\mathrm{PJ})^h}{(\Box A \Rightarrow \Box B, \Box C), (\boxdot A \Rightarrow B \mid \boxdot A \Rightarrow C)\ \triangleright\ (\boxdot A \Rightarrow B \vee C)}\,(\mathrm{w})\triangleright}{(\Box A \Rightarrow \Box B, \Box C)\ \triangleright\ (\boxdot A \Rightarrow B \vee C)}\,(\mathrm{v}^i)^h$$

Proposition 1. *Let* L *be a mono-extensible modal logic.*

(a) All the core modal rules are L*-sound.*
(b) If L *is irreflexive (in particular, a* GL*-extension), then* $(\mathrm{v}^i)^h$ *is* L*-sound.*
(c) If L *is reflexive (equivalently, an* $\mathsf{S4}$*-extension), then* $(\mathrm{v}^r)^h$ *and* $(\mathrm{AC}_\Box)$ *are* L*-sound.*

Proof. Many parts of this proof are exactly as in the extensible case considered in [9]. It remains only to check the soundness of the Visser rules $(\mathrm{v}^i)^h$ and $(\mathrm{v}^r)^h$. For the former, it is sufficient to show $I^\Box(G \mid \Box\Gamma \Rightarrow \Box\Delta) \vdash_{\mathsf{L}} I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$. Suppose that $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$ for some substitution σ. If $\Delta = \emptyset$, then $\nvdash_{\mathsf{L}} \sigma I^\Box(G)$ and since $\vdash_{\mathsf{L}} \Box\neg\Box A \leftrightarrow \Box\bot$ for any formula A, easily $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \Box\Gamma \Rightarrow)$. Suppose then that $\Delta \neq \emptyset$. Since L has the FMP, let K be a finite L-model refuting $\sigma I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$ and let F be the frame of K. L is mono-extensible and irreflexive, so F^i is also an L-frame. Consider a model on the frame F^i for which the forcing in all nodes except the root is the same as in K, and no variables are forced at the root. $\Box I(\Box\sigma\Gamma \Rightarrow \Box\sigma\Delta)$ and $\Box I(\sigma S)$ are refuted at the root for all $S \in G$, so $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \Box\Gamma \Rightarrow \Box\Delta)$ and we are done. Note that the extra boxes in the interpretation $I^\Box$ of hypersequents is essential here, since we cannot conclude that G is not forced at the root, but we can for $I^\Box(G)$.

For (v^r), we show $I^\Box(G \mid \Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta) \vdash_{\mathsf{L}} I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$. Suppose that $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$ for some substitution σ. If $\Delta = \emptyset$, then $\nvdash_{\mathsf{L}} \sigma I^\Box(G)$ and since $\vdash_{\mathsf{L}} \Box\neg(A \leftrightarrow \Box A) \leftrightarrow \Box\bot$ for any formula A, easily $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \Gamma \equiv \Box\Gamma \Rightarrow)$. Suppose then that $\Delta \neq \emptyset$. Since L has the FMP, let K be a finite L-model refuting $\sigma I^\Box(G \mid \{\boxdot\Gamma \Rightarrow C : C \in \Delta\})$ and let F be the frame of K. L is mono-extensible and reflexive, so F^r is an L-frame with a reflexive root r. By the reflexivity of r and since K forces $\sigma(\boxdot\Gamma)$, r forces $\sigma(A) \leftrightarrow \sigma(\Box A)$ for all $A \in \Gamma$. Hence r forces $\bigwedge\{\sigma(A) \leftrightarrow \sigma(\Box A) : A \in \Gamma\}$ but refutes $\sigma(\bigvee \Box\Delta)$. So $\nvdash_{\mathsf{L}} \sigma I^\Box(G \mid \Gamma \equiv \Box\Gamma \Rightarrow \Box\Delta)$. □

Notice that we have considered here only logics that are either reflexive or irreflexive, meaning that logics such as $\mathsf{K4.2}$ lacking these properties are currently beyond our scope (although we believe that very similar methods should suffice for such cases).

We now turn our attention to establishing completeness for fitting calculi, restricting our attention (at least to start with) to the single-conclusion case. First, we show that L-derivable sgh-rules are also GAML-derivable.

Lemma 2. *Let* L *be a mono-extensible modal logic and let* GAML *be* L*-fitting. If* $\vdash_{\mathsf{L}} \mathtt{R}$*, then* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any sgh-rule* $\mathtt{R}$*.*

Proof. Let $\mathtt{R} = (\mathcal{G} \rhd \mathcal{H})$ where $|\mathcal{H}| \leq 1$ and suppose that $\vdash_{\mathsf{L}} \mathtt{R}$. If $\mathcal{H}$ is $\{H\}$, or taking H to be $(\Rightarrow)$ if $\mathcal{H} = \emptyset$, then $\bigwedge_{G \in \mathcal{G}} I^{\Box}(G) \vdash_{\mathsf{L}} I^{\Box}(H)$. But then since we are above $\mathsf{K4}$, using the modal deduction theorem:

$$\vdash_{\mathsf{L}} \bigwedge_{G \in \mathcal{G}} \boxdot I^{\Box}(G) \to I^{\Box}(H)$$

GAML is L-fitting, so by repeated applications of $(\mathrm{PJ})^h$, $\vdash_{\mathsf{GAML}} \mathcal{G} \rhd \mathcal{H}$ as required. □

Our task now is to reduce the L-admissibility of a gh-rule to the admissibility of more manageable gh-rules. We introduce the following notions:

Definition 4. *A gh-rule* $\mathtt{R} = (\mathcal{G} \rhd \mathcal{H})$ *is:*

- modal-irreducible *if* $\mathcal{G}$ *contains only variables and boxed variables.*
- modal-semi-irreducible *if* $\mathcal{G}$ *contains only variables and boxed variables, and on the left of sequents possibly also equivalences of the form* $p \leftrightarrow \Box p$.
- full *with respect to a set of rules* X *if whenever* $\mathtt{R}_1, \ldots, \mathtt{R}_n/\mathtt{R}$ *is an instance of a rule in* X*, then* $\mathtt{R}_i \subseteq \mathtt{R}$ *for some* $i \in \{1, \ldots, n\}$ *(i.e. applying a rule in* X *backwards to* $\mathtt{R}$ *adds no new sequents to the gh-rule).*

It is an easy task (see e.g. [9]) – essentially following from the soundness of the usual logical rules for modal logics – to show that the left logical rules are all L-invertible. Moreover, each such rule (working upwards) removes an occurrence of a logical connective from a sequent in a hypersequent on the left. Hence, if we define the complexity of a sequent as the multiset of complexities (number of symbols) of its formulas, and the complexity of a gh-rule as the multiset of complexities of its sequents, then it is a standard inductive proof to show the following:

Lemma 3. *Each* L*-admissible gh-rule can be derived from an* L*-admissible modal-irreducible gh-rule using the left logical rules.*

But now notice that there is a finite number of different semi-modal-irreducible sequents built from a fixed set of variables. Hence applying any number of rules with the subformula property backwards to a modal-irreducible gh-rule will terminate with gh-rules full with respect to that set.

Lemma 4. *Let* $X \subseteq \{(\mathrm{V}^i)^h, (\mathrm{V}^r)^h, (\mathrm{AC}), (\mathrm{AC}_{\Box}), (\wedge \Rightarrow)\rhd, (\to \Rightarrow)\rhd\}$*. Then every modal-irreducible* L*-admissible gh-rule can be derived using* X *from a set of semi-modal-irreducible* L*-admissible gh-rules that are full with respect to* X*. If* X *does not contain* $(\mathrm{AC}_{\Box})$*, then these gh-rules are modal-irreducible.*

For the main part of the proof, we again consider only irreflexive and reflexive logics (and single-conclusion gh-rules), treated by the following two theorems:

Theorem 3. *If* GAML *is* L*-fitting for a mono-extensible irreflexive modal logic* L*, then* $\vdash\!\!\sim_{\mathsf{L}} \mathtt{R}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any sgh-rule* $\mathtt{R}$.

Proof. The right-to-left direction follows from the definition of L-fitting and Proposition 1. For the left-to-right direction, it is sufficient by Lemma 4 to assume that $\mathtt{R} = (\mathcal{G} \rhd \mathcal{H})$ is an L-admissible modal-irreducible sgh-rule that is full with respect to (v^i) and (AC). Suppose now that:

$$\mathcal{G} = (G_1, \ldots, G_n) \quad \text{and} \quad G_i = (S^i_1 \mid \ldots \mid S^i_{m_i}) \quad \text{where} \quad S^i_j = (\Gamma^i_j \Rightarrow \Delta^i_j).$$

Let $A = \bigwedge_{i=1}^n I^\Box(G_i) = \bigwedge_{i=1}^n \bigvee_{j=1}^{m_i} \Box I(S^i_j)$. If A is inconsistent, then $\vdash_{\mathsf{GAML}} \mathcal{G} \rhd \mathcal{H}$ follows immediately by Lemma 2. We define:

$$X_{j_1,\ldots,j_n} = \{S^1_{j_1}, \ldots, S^n_{j_n}\} \quad \text{and} \quad C = \bigvee_{j_1 \leq m_1, \ldots, j_n \leq m_n} \bigwedge_{S \in X_{j_1,\ldots,j_n}} \Box I(S).$$

and observe that by distributivity: $\vdash_{\mathsf{L}} C \leftrightarrow A$.

Now we come to the crucial point of the proof. Each irreflexive logic with the FMP contains GL. Suppose that we can show that $\bigwedge_{S \in X_{j_1,\ldots,j_n}} I(S)$ is GL-projective or GL-inconsistent for each $j_1 \leq m_1, \ldots, j_n \leq m_n$. For $\mathcal{H} = \{H\}$ or taking H as $\Rightarrow$ if $\mathcal{H} = \emptyset$, it follows by Lemma 1 (d), that $C \vdash_{\mathsf{L}} I^\Box(H)$ iff $C \vdash\!\!\sim_{\mathsf{L}} I^\Box(H)$. So since $\vdash_{\mathsf{L}} C \leftrightarrow A$ and $A \vdash\!\!\sim_{\mathsf{L}} I^\Box(H)$, we get $\vdash_{\mathsf{L}} \mathcal{G} \rhd \mathcal{H}$. But then by Lemma 2, $\vdash_{\mathsf{GAML}} \mathcal{G} \rhd \mathcal{H}$ as required. Note that the fact that $\mathcal{H}$ consists of at most one hypersequent plays a crucial role here.

Hence we have proved the theorem once we have shown that each $\bigwedge_{S \in X_{j_1,\ldots,j_n}} I(S)$ is either GL-projective or GL-inconsistent. To achieve this we make use of the result established in [9], that a modal irreducible *gs-rule* that is full with respect to (v^i) and (AC) is either projective or inconsistent. I.e. it is sufficient to show that $X_{j_1,\ldots,j_n} \rhd \mathcal{H}$ is a modal irreducible gs-rule that is full with respect to (v^i) and (AC). The rest of this proof will consist of a proof of this fact.

We call a set $X_{j_1,\ldots,j_n}$ *minimal* if it does not contain a proper subset $X_{h_1,\ldots,h_n}$. It suffices to establish the modal irreducibility and fullness with respect to (v^i) and (AC) only for the gs-rules $X_{j_1,\ldots,j_n} \rhd \mathcal{H}$ for which $X_{j_1,\ldots,j_n}$ is minimal. For suppose that there is a $X_{h_1,\ldots,h_n} \subset X_{j_1,\ldots,j_n}$ for which $I(X_{h_1,\ldots,h_n})$ is GL-projective or GL-inconsistent. Then $\vdash_{\mathsf{GAML}} X_{h_1,\ldots,h_n} \rhd \mathcal{H}$, and, since $X_{h_1,\ldots,h_n} \subset X_{j_1,\ldots,j_n}$, $\vdash_{\mathsf{GAML}} X_{j_1,\ldots,j_n} \rhd \mathcal{H}$.

Let us fix a minimal $\mathcal{D} = X_{j_1,\ldots,j_n}$ for some $j_1 \leq m_1, \ldots, j_n \leq m_n$. Clearly $\mathcal{D} \rhd \mathcal{H}$ is modal-irreducible. The following two claims establish the fullness of $\mathcal{D} \rhd \mathcal{H}$ with respect to (v^i) and (AC), which completes the proof.

Claim. $\mathcal{D} \rhd \mathcal{H}$ is full with respect to (v^i).

Proof. Suppose that $\mathcal{D}$ contains a sequent $(\Box\Gamma \Rightarrow \Box\Delta)$. Then $\mathcal{G}$ contains a hypersequent $(G \mid \Box\Gamma \Rightarrow \Box\Delta)$. We have to show that it contains a sequent $(\boxdot\Gamma \Rightarrow A)$ for some $A \in \Delta$. By the fullness of $\mathcal{G} \rhd \mathcal{H}$ with respect to $(\mathrm{v}^i)^h$, it follows that $\mathcal{G}$ contains the hypersequent $(G \mid \{\boxdot\Gamma \Rightarrow A \mid A \in \Delta\})$ (just G if $\Delta = \emptyset$). By the definition of $\mathcal{D} = X_{j_1,\ldots,j_n}$ it follows that either $(\boxdot\Gamma \Rightarrow A)$ belongs to $\mathcal{D}$, in which case we are done, or there is a sequent S in G that belongs to $\mathcal{D}$. But it is not hard to see that in this case there is a set $X_{h_1,\ldots,h_n}$, corresponding to a disjunct of C, that is the result of replacing $(\Box\Gamma \Rightarrow \Box\Delta)$ in $\mathcal{D}$ by S. But then $X_{h_1,\ldots,h_n}$ is a proper subset of $\mathcal{D}$, contradicting the minimality of $\mathcal{D}$. Observe that we use here the fact that no hypersequent in $\mathcal{G}$, being just a set of sequents, can contain the same sequent twice.

Claim. $\mathcal{D} \triangleright \mathcal{H}$ is full with respect to (AC).

Proof. The proof is similar to the proof of the claim above, but let us spell it out nevertheless. Suppose that $\mathcal{D}$ contains the sequents $(\Gamma, A \Rightarrow \Delta)$ and $(\Pi \Rightarrow A, \Sigma)$. We have to show that it contains the sequent $(\Gamma, \Pi \Rightarrow \Sigma, \Delta)$. Observe that $\mathcal{G}$ has to contain hypersequents of the form $(G \mid \Gamma, A \Rightarrow \Delta)$ and $(H \mid \Pi \Rightarrow A, \Sigma)$ for some hypersequents G and H. By the fullness of $\mathcal{G} \triangleright \mathcal{H}$ with respect to $(\mathrm{AC})^h$, it follows that if $\mathcal{D}$ does not contain the sequent $(\Gamma, \Pi \Rightarrow \Sigma, \Delta)$ it has to contain a sequent S from G or H. In this case, replacing the $(\Gamma, A \Rightarrow \Delta)$ in $\mathcal{D}$ by S in case S occurs in G and replacing $(\Pi \Rightarrow A, \Sigma)$ by S otherwise, we obtain a set $X_{h_1,\ldots,h_n}$, corresponding to a disjunct of C, that is a proper subset of $\mathcal{D}$, contradicting the minimality of $\mathcal{D}$. □

Theorem 4. *If* GAML *is* L*-fitting for a mono-extensible reflexive modal logic* L*, then* $\vdash_{\mathsf{L}} \mathtt{R}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any sgh-rule* R.

Proof. Since the reasoning is similar to the completeness proof for irreflexive logics given above, we just explain the points of divergence and leave the details to the reader. In this case, for the left-to-right direction, it is sufficient by Lemma 4 to assume that $\mathtt{R} = (\mathcal{G} \triangleright \mathcal{H})$ is a modal-semi-irreducible L-admissible sgh-rule that is full with respect to $(\mathrm{V}^r)^h$, $(\mathrm{AC})^h$, $(\mathrm{AC}_\Box)^h$, $(\wedge\Rightarrow)\triangleright^h$, and $(\rightarrow\Rightarrow)\triangleright^h$, and obtained by applying these rules (backwards) to a modal-irreducible gh-rule. We then define $X_{j_1,\ldots,j_n}$, C, and $\mathcal{D}$ as in the irreflexive case. This time, since each reflexive extension of K4 contains S4, it is sufficient to show that each $\bigwedge_{S \in X_{j_1,\ldots,j_n}} I(S)$ is either S4-projective or S4-inconsistent.

To achieve this we make use of the result established in [9], that a modal-semi-irreducible *gs-rule* that is full with respect to $(\mathrm{V}^r)^h$, $(\mathrm{AC})^h$, $(\mathrm{AC}_\Box)^h$, $(\wedge\Rightarrow)\triangleright^h$, and $(\rightarrow\Rightarrow)\triangleright^h$, is either S4-projective or S4-inconsistent. I.e. it is sufficient to show that $\mathcal{D} \triangleright \mathcal{H}$ is a modal-semi-irreducible gs-rule that is full with respect to $(\mathrm{V}^r)^h$, $(\mathrm{AC})^h$, $(\mathrm{AC}_\Box)^h$, $(\wedge\Rightarrow)\triangleright^h$, and $(\rightarrow\Rightarrow)\triangleright^h$. The proofs of these facts are similar to the proofs of the claims in the previous proof, and are left to the reader. □

The obvious question remaining here (apart from extending beyond the reflexive and irreflexive cases) is what happens in the case of multiple-conclusion rules. We just give a partial answer here, leaving the general case for further investigation. First, we recall from [10] that L has *essentially single-conclusion* admissible rules if whenever $\Gamma \vdash_{\mathsf{L}} \Delta$, there exists $A \in \Delta \cup \{\bot\}$ such that $\Gamma \vdash_{\mathsf{L}} A$. Jeřábek has shown the following using the notion of filtering unification investigated by Ghilardi and Sacchetti in [6].

Theorem 5 ([10]). *Every extension of* K4.2 *has essentially single-conclusion admissible rules. Moreover, any normal extension of* K4.1 *with essentially single-conclusion admissible rules is an extension of* K4.2.

Corollary 1. *If* GAML *is* L*-fitting for a mono-extensible reflexive or irreflexive extension* L *of* K4.2*, then* $\vdash_{\mathsf{L}} \mathtt{R}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any gh-rule* R.

Let us note finally for this section that concrete systems for admissiblity can be defined for mono-extensible logics such as S4.2, GL.3, etc. by adding to our core set of rules any kind of calculus for derivability in these logics. In particular, hypersequent calculi can be developed for many of these cases, but we omit the details here for space reasons.

Initial GS-Rules, Structural Rules, Anti-Cut Rule, Projection Rule: *as in the core modal rules*.

Logical Rules: *as in the core modal rules for* $\bot$, $\wedge$, *and* $\vee$, *plus*:

$$\frac{\mathcal{G} \rhd (\Gamma, A \to B \Rightarrow A, \Delta), \mathcal{H} \quad \mathcal{G} \rhd (\Gamma, B \Rightarrow \Delta), \mathcal{H}}{\mathcal{G} \rhd (\Gamma, A \to B \Rightarrow \Delta), \mathcal{H}} \rhd(\to\Rightarrow)^i \qquad \frac{\mathcal{G} \rhd (\Gamma, A \Rightarrow B), \mathcal{H}}{\mathcal{G} \rhd (\Gamma \Rightarrow A \to B, \Delta), \mathcal{H}} \rhd(\Rightarrow\to)^i$$

$$\frac{\mathcal{G}, (\Gamma, B \Rightarrow \Delta), (\Gamma, A \to B \Rightarrow A, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \to B \Rightarrow \Delta) \rhd \mathcal{H}} (\to)\rhd$$

$$\frac{\mathcal{G}, (\Gamma \Rightarrow p, \Delta), (p, A \Rightarrow B) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma \Rightarrow A \to B, \Delta) \rhd \mathcal{H}} (\Rightarrow\to)\rhd^i \qquad \frac{\mathcal{G}, (\Gamma, p \to q \Rightarrow \Delta), (p \Rightarrow A), (B \Rightarrow q) \rhd \mathcal{H}}{\mathcal{G}, (\Gamma, A \to B \Rightarrow \Delta) \rhd \mathcal{H}} (\to\Rightarrow)\rhd^i$$

where p and q do not occur in $\mathcal{G}$, $\mathcal{H}$, Γ, and Δ in $(\to\Rightarrow)\rhd^i$, $(\Rightarrow\to)\rhd^i$.

Visser Rule

$$\frac{[\mathcal{G}, (\Gamma \Rightarrow \Delta), (\Gamma \Rightarrow A) \rhd \mathcal{H}]_{A \in \Delta} \quad [\mathcal{G}, (\Gamma \Rightarrow \Delta) \rhd (\Gamma^\Pi, \Pi \Rightarrow \Delta), \mathcal{H}]_{\emptyset \neq \Pi \subseteq \Gamma_\Delta}}{\mathcal{G}, (\Gamma \Rightarrow \Delta) \rhd \mathcal{H}} \text{(V)}$$

where Γ contains only implications, and:

1. $\Gamma^\Pi = \{A \to B \in \Gamma : A \notin \Pi\}$. 2. $\Gamma_\Delta = \{A \notin \Delta : \exists B\,(A \to B) \in \Gamma\}$.

Fig. 4. The Calculus GAMI

4 Intermediate Logics

We turn our attention now to intermediate logics, recalling first the result of [8] that if an intermediate logic L admits the following *Visser rules*, then they form a basis for the admissible rules of L:

$$(V_n)\ \ (C \to (A_{n+1} \vee A_{n+2})) \vee D \,/\, (\bigvee_{j=1}^{n+2} C \to A_j) \vee D$$

for $n = 1, 2, \ldots$, where $C = \bigwedge_{i=1}^{n}(A_i \to B_i)$. In some cases, such as Gödel-Dummett logic LC, the Visser rules (and hence all admissible rules) are derivable. Here we consider some logics where this does not happen: in particular, de Morgan (or Jankov) logic KC, axiomatized by adding the axiom $\neg A \vee \neg\neg A$ to IPC, and the family of logics with Kripke models of bounded cardinality BC_n for $n = 1, 2, \ldots$ (noting that for $n = 1, 2$, the Visser rules are in fact derivable).

We also recall Ghilardi's useful characterization of IPC-projective formulas.

Theorem 6 (Ghilardi [4]). *For Kripke models* $K_1, \ldots, K_n$, *let* $(\sum_i K_i)'$ *denote the Kripke model obtained by attaching one new node below all nodes in* $K_1, \ldots, K_n$ *where no variables are forced. A class of Kripke models* $\mathcal{K}$ *has the* extension property *if for every finite family of models* $K_1, \ldots, K_n \in \mathcal{K}$, *there is a variant of* $(\sum_i K_i)'$ *in* $\mathcal{K}$. *A formula is* IPC *-projective iff its class of Kripke models has the extension property.*

Figure 4 displays the gs-rule calculus GAMI for Intuitionistic Logic of [9]. In this case it is the (non-invertible on the right) implication rules that use new variables on the left.

Theorem 7 ([9]). $\vdash_{\mathsf{GAMI}} \mathtt{R}$ *iff* $\vdash_{\mathsf{IPC}} \mathtt{R}$ *for any gs-rule* R.

$$\frac{\mathcal{G}, (G \mid \Gamma, A \Rightarrow \Delta), (H \mid \Pi \Rightarrow A, \Sigma)(G \mid H \mid \Gamma, \Pi \Rightarrow \Sigma, \Delta) \rhd \mathcal{H}}{\mathcal{G}, (G \mid \Gamma, A \Rightarrow \Delta), (H \mid \Pi \Rightarrow A, \Sigma) \rhd \mathcal{H}} \ (\text{AC})^h$$

$$\frac{\mathcal{G}, G \rhd (I(G) \Rightarrow I(H)), \mathcal{H}}{\mathcal{G}, G \rhd \mathcal{H}} \ (\text{IPJ}) \quad \text{where } H \in \mathcal{H} \cup \{\Rightarrow\}$$

$$\frac{\mathcal{G}, (G \mid \{\Gamma \Rightarrow A\}_{A \in \Delta}) \rhd \mathcal{H} \quad [\mathcal{G} \rhd (\Gamma^{\Pi}, \Pi \Rightarrow \Delta), \mathcal{H}]_{\Pi \subseteq \Gamma_\Delta}}{\mathcal{G}, (G \mid \Gamma \Rightarrow \Delta) \rhd \mathcal{H}} \ (\text{V})^h$$

where Γ contains only implications, and:
1. $\Gamma^{\Pi} = \{A \to B \in \Gamma : A \notin \Pi\}$. 2. $\Gamma_\Delta = \{A \notin \Delta : \exists B\,(A \to B) \in \Gamma\}$.

Fig. 5. Additional Rules for Intermediate Logics

Just as we stepped from extensible to mono-extensible modal logics, so we can step here from a calculus for IPC to calculi for intermediate logics admitting the Visser rules. Note, however, that we require a slightly different (more usual) interpretation of hypersequents. For an intermediate logic L and hypersequent G, we write $\vdash_{\mathsf{L}} G$ iff $\vdash_{\mathsf{L}} I(G)$ where $I(G) = \bigvee_{S \in G} I(S)$. Also, a gh-rule $\mathtt{R} = (G_1, \ldots, G_n \rhd H_1, \ldots, H_m)$ is L-*admissible*, written $\sim_{\mathsf{L}} \mathtt{R}$, iff $\{I(G_i)\}_{i=1}^n \vdash_{\mathsf{L}} \{I(H_j)\}_{j=1}^m$ and L-*derivable*, written $\vdash_{\mathsf{L}} \mathtt{R}$, iff $\bigwedge_{i=1}^n I(G_i) \vdash_{\mathsf{L}} \bigvee_{j=1}^m I(H_j)$. It will also be helpful to restrict the notion of an sgh-rule a little further to an *ssgh-rule*: an sgh rule where not only is there at most one hypersequent on the right, but also this hypersequent consists of just one sequent. Essentially, the reason for this is that completeness results for hypersequent calculi for intermediate logics given in the literature (see e.g. [3]) are typically restricted to sequents rather than hypersequents.

Definition 5. *A calculus* GAML *is* L*-fitting for an intermediate logic* L *if:*

(1) GAML *extends the* core intermediate rules*: gh-versions of the initial gs-rules, structural rules, and logical rules of* GAMI*, and the additional rules of Fig. 5.*
(2) If $\vdash_{\mathsf{L}} S$*, then* $\vdash_{\mathsf{GAML}} \rhd S$ *for any sequent* S*.*
(3) If $\vdash_{\mathsf{GAML}} \mathtt{R}$*, then* $\sim_{\mathsf{L}} \mathtt{R}$*.*

Lemma 5. *The core intermediate rules are* L*-sound for every intermediate logic* L *admitting the Visser rules.*

Proof. Let L be an intermediate logic admitting the Visser rules. We just consider $(\text{V})^h$ since other proofs are very similar to those for GAMI in [9]. Suppose that σ is an L-unifier for $I(H)$ for all $H \in \mathcal{G}$ and $I(G \mid \Gamma \Rightarrow \Delta)$, where $\Delta = \{A_1, \ldots, A_n\}$. Using the right set of premises, σ is an L-unifier for $I(\Gamma^{\Pi}, \Pi \Rightarrow \Delta)$ for all $\Pi \subseteq \Gamma_\Delta$. It suffices now to show that σ is an L-unifier for $I(G \mid \{\Gamma \Rightarrow A\}_{A \in \Delta})$. Suppose, arguing contrapositively, that this is not the case. Then there exists a countermodel of L for $I(\sigma(G)) \vee \bigvee_{A \in \Delta} I(\sigma(\Gamma) \Rightarrow \sigma(A))$. This implies that for every $A \in \Delta$ there are countermodels K_A such that K_A is a model of L, $K_A \Vdash \sigma(\bigwedge \Gamma)$, and $K_A \nVdash \sigma(A)$. Since L admits the Visser rules, there is a variant K of $(\sum_{A \in \Delta} K_A)'$ that is a model of L. Let $\Pi = \{D \in \Gamma_\Delta : K \Vdash \sigma(D)\}$. Observe that for all $B \to C \in \Gamma$ such that $B \notin \Pi$, either $B \in \Delta$ or $K \nVdash \sigma(B)$. Note also that $B \in \Delta$ implies $K \nVdash \sigma(B)$. Hence

for all $B \notin \Pi$ it follows that $K \nVdash \sigma(B)$, and so $K \Vdash \sigma(B \to C)$. It follows that $K \Vdash \sigma(\bigwedge(\Gamma^{\Pi} \cup \Pi))$. Thus $K \Vdash \sigma(\bigvee \Delta)$, a contradiction. □

The difference between the rules $(\mathrm{V})^h$ and (V) is essentially due to the fact that IPC has the disjunction property, while intermediate logics in general do not.

The core set of rules given above can be extended on the right to obtain proof systems for admissibility in various intermediate logics. In particular, we can make use of hypersequent calculi provided for KC and BC_n ($n = 1, 2, \ldots$) in [3], to obtain:

- GAMKC consists of the core intermediate rules plus:

$$\frac{\mathcal{G} \rhd (G \mid \Gamma_1 \Rightarrow \Delta_1 \mid \Gamma_2 \Rightarrow \Delta_2 \mid \Gamma_1, \Gamma_2 \Rightarrow)}{\mathcal{G} \rhd (G \mid \Gamma_1 \Rightarrow \Delta_1 \mid \Gamma_2 \Rightarrow \Delta_2)}\ (\mathrm{J})$$

- $\mathsf{GAMBC_n}$ for $n = 1, 2, \ldots$ consists of the core intermediate rules plus:

$$\frac{[\mathcal{G} \rhd (G \mid \Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_{n+1} \Rightarrow \Delta_{n+1} \mid \Gamma_i, \Gamma_j \Rightarrow \Delta_i)]_{1 \le i < j \le n+1}}{\mathcal{G} \rhd (G \mid \Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_{n+1} \Rightarrow \Delta_{n+1})}\ (\mathrm{BC}_n)$$

Corollary 2. GAMKC *is* KC*-fitting and* $\mathsf{GAMBC_n}$ *is* BC_n*-fitting for* $n = 1, 2, \ldots$.

To prove completeness for L-fitting systems GAML for intermediate logics L admitting the Visser rules, we proceed similarly to the case of mono-extensible modal logics. First we can show, exactly as in Lemma 2 (except replacing the application of the modal deduction theorem with the usual deduction theorem), that L-derivable ssgh-rules are also GAML-derivable.

Lemma 6. *Let* L *be an intermediate logic admitting the Visser rules and let* GAML *be* L*-fitting. If* $\vdash_{\mathsf{L}} \mathtt{R}$*, then* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any ssgh-rule* R.

As for Lemmas 3 and 4, applying the invertible left logical rules backwards reduces any gh-rule to a gh-rule of a certain form (in this case with variable implications on the left), and then applying the rules $(\mathrm{V})^h$, $(\to)^{\rhd h}$, and $(\mathrm{AC})^h$ exhaustively backwards terminates with a set of gh-rules full with respect to these rules.

Lemma 7. *A gh-rule* $\mathcal{G} \rhd \mathcal{H}$ *is* implication-irreducible *if all sequents in* $\mathcal{G}$ *contain only variables on the right and variables and variable implications on the left. Every admissible gh-rule is* GAML*-derivable from admissible implication-irreducible gh-rules that are full with respect to* $(\mathrm{V})^h$*,* $(\to)^{\rhd h}$*, and* $(\mathrm{AC})^h$.

The completeness theorem is then established similarly to the proof for IPC in [9], the main complication being that (as in the mono-extensible modal case) we now have to take care of all the different disjuncts occurring in hypersequents on the left.

Theorem 8. *For any intermediate logic* L *admitting the Visser rules and* L*-fitting calculus* GAML, $\vdash\!\!\sim_{\mathsf{L}} \mathtt{R}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any ssgh-rule* R.

Proof. The right-to-left direction follows directly from Lemma 5. For the left-to-right direction, it is sufficient to assume (proceeding exactly as in the IPC-case) that

$\mathtt{R} = (\mathcal{G} \rhd \mathcal{H})$ is an L-admissible implication-irreducible gh-rule that is full with respect to $(\mathrm{V})^h$, $(\rightarrow)\rhd^h$, and $(\mathrm{AC})^h$. The proof is similar to the completeness proofs for the modal logics given above, but since some details are essentially different we will sketch the proof for intermediate logics briefly.

Define C and $X_{j_1,\ldots,j_n}$ as in the modal completeness proofs above, but without boxes. By Lemma 1 (d), if each $I(X_{j_1,\ldots,j_n})$ is IPC-projective or IPC-inconsistent, then $\vdash_{\mathsf{L}} \mathcal{G} \rhd \mathcal{H}$. But then by Lemma 6, $\vdash_{\mathsf{GAML}} \mathcal{G} \rhd \mathcal{H}$. It is then sufficient to show that each consistent $\bigwedge_{X_{j_1,\ldots,j_n}} I(S)$ is IPC-projective or derives $I(\mathcal{H})$. Recall that $X_{j_1,\ldots,j_n}$ is called minimal if there is no $X_{h_1,\ldots,h_n}$ that is a proper subset of $X_{j_1,\ldots,j_n}$. Reasoning as in the modal case, it suffices to consider only minimal sets. Let $\mathcal{D}$ denote a minimal set $X_{j_1,\ldots,j_n}$. To show that $\mathcal{D} \rhd \mathcal{H}$ has the mentioned properties, we make use of the result established in [9], that an implication-irreducible *gs-rule* that is full with respect to $(\mathrm{V})^h$, $(\rightarrow)\rhd^h$, and $(\mathrm{AC})^h$ is either projective or derives $I(\mathcal{H})$. I.e. it is sufficient to show that $\mathcal{D} \rhd \mathcal{H}$ is an implication-irreducible gs-rule that is full with respect to $(\mathrm{V})^h$, $(\rightarrow)\rhd^h$, and $(\mathrm{AC})^h$. Proofs of these facts are similar to the claims in the completeness proofs for modal logics. We will only treat $(\mathrm{V})^h$, leaving the other cases to the reader.

Claim. $\mathcal{D} \rhd \mathcal{H}$ is full with respect to the rule $(\mathrm{V})^h$.

Proof. Suppose that $\mathcal{D}$ contains a sequent $(\Gamma \Rightarrow \Delta)$, where Γ contains only implications. Thus $\mathcal{G}$ contains a hypersequent $(G \mid \Gamma \Rightarrow \Delta)$ for some hypersequent G. By the fullness of $(\mathcal{G} \rhd \mathcal{H})$ with respect to $(\mathrm{V})^h$, it follows that either $\mathcal{H}$ contains $(\Gamma^{\Pi}, \Pi \Rightarrow \Delta)$ for some $\Pi \subseteq \Gamma_\Delta$, or $\mathcal{D}$ contains a sequent $(\Gamma \Rightarrow A)$ for some $A \in \Delta$, or $\mathcal{D}$ contains a sequent S of G. In the first two cases we are done. In the last case, by replacing $(\Gamma \Rightarrow \Delta)$ by S in $\mathcal{D}$, we obtain a set $X_{h_1,\ldots,h_n}$, corresponding to a disjunct of C, that is a proper subset of $\mathcal{D}$, contradicting the minimality of $\mathcal{D}$. □

Corollary 3. *For* $\mathsf{L} \in \{\mathsf{KC}, \mathsf{BC}_1, \mathsf{BC}_2, \ldots\}$*:* $\mathtt{R} \in \mathsf{L}$ *iff* $\vdash_{\mathsf{GAML}} \mathtt{R}$ *for any ssgh-rule* $\mathtt{R}$.

As in the modal case, there exist essentially single-conclusion logics where the preceding corollary extends to multiple-conclusion rules; indeed, we conjecture that this is the case for all extensions of KC admitting the Visser rules.

References

1. Avron, A.: A constructive analysis of RM. Journal of Symbolic Logic 52(4), 939–951 (1987)
2. Chagrov, A., Zakharyaschev, M.: Modal Logic. Oxford University Press, Oxford (1996)
3. Ciabattoni, A., Ferrari, M.: Hypersequent calculi for some intermediate logics with bounded Kripke models. Journal of Logic and Computation 11(2), 283–294 (2001)
4. Ghilardi, S.: Unification in intuitionistic logic. Journal of Symbolic Logic 64(2), 859–880 (1999)
5. Ghilardi, S.: Best solving modal equations. Annals of Pure and Applied Logic 102(3), 184–198 (2000)
6. Ghilardi, S., Sacchetti, L.: Filtering unification and most general unifiers in modal logic. Journal of Symbolic Logic 69(3), 879–906 (2004)
7. Iemhoff, R.: On the admissible rules of intuitionistic propositional logic. Journal of Symbolic Logic 66(1), 281–294 (2001)

8. Iemhoff, R.: Intermediate logics and Visser's rules. Notre Dame Journal of Formal Logic 46(1), 65–81 (2005)
9. Iemhoff, R., Metcalfe, G.: Proof theory for admissible rules. (submitted), http://www.math.vanderbilt.edu/people/metcalfe/publications
10. Jerábek, E.: Admissible rules of modal logics. Journal of Logic and Computation 15, 411–431 (2005)
11. Metcalfe, G., Olivetti, N., Gabbay, D.: Proof Theory for Fuzzy Logics. Springer Series in Applied Logic, vol. 36. Springer, Heidelberg (to appear, 2009)
12. Rozière, P.: Regles Admissibles en calcul propositionnel intuitionniste. PhD thesis, Université Paris VII (1992)
13. Rybakov, V.: Admissibility of Logical Inference Rules. Elsevier, Amsterdam (1997)

Light Linear Logic with Controlled Weakening

Max Kanovich

Dept. of Computer Science, Queen Mary, University of London
mik@dcs.qmul.ac.uk

Abstract. Starting from Girard's seminal paper on light linear logic (LLL), a number of works investigated on systems derived from linear logic to capture polynomial time computation within the computation-as-cut-elimination paradigm.

The original syntax of LLL is too complicated, mainly because one has to deal with sequents which not just consist of formulas but also of 'blocks' of formulas. We circumvent the complications of 'blocks' by introducing a new modality ∇ which is exclusively in charge of 'additive blocks'. The most interesting feature of this purely multiplicative ∇ is the possibility of the second-order encodings of additive connectives.

The resulting system (with the traditional syntax), called Easy-LLL, is still powerful to represent any deterministic polynomial time computations in purely logical terms. Unlike the original LLL, Easy-LLL admits polynomial time strong normalization, namely, cut elimination terminates in a *unique way* in polytime by any choice of cut reduction strategies.

1 Introduction and Summary

Girard[8] has introduced light linear logic (LLL) as a refinement of the 'proofs-as-programs' paradigm to polynomial-time computation.

The basic idea of the representability of computation in *purely logical terms* is the following.

Let every (binary) number m be associated with a certain cut-free proof π_m of a fixed formula, say $\mathtt{Num}$. A function f mapping from numbers into numbers is *represented by a proof* $\widetilde{\Pi}$ of the sequent $\vdash \mathtt{Num}^{\perp}; \mathtt{Num}$, if for any number m, a cut elimination procedure applied to the proof:[1]

$$\dfrac{\dfrac{\pi_m}{\vdash \mathtt{Num}} \qquad \dfrac{\widetilde{\Pi}}{\vdash \mathtt{Num}^{\perp}; \mathtt{Num}}}{\vdash \mathtt{Num}}\ (\mathrm{cut})$$

results in a cut-free proof $\pi_{f(m)}$ associated with $f(m)$. An implicit *fairness condition* is that the number of cut reductions must be in 'polynomial' accordance with the time complexity of f.

Starting from Girard's seminal paper on light linear logic[8], a number of works investigated on systems derived from linear logic and corresponding to

[1] To contract a number of inference rules, we will prefer one-side calculi. As usual in linear logic, *linear negation* $A^{\perp}$ is used as an abbreviation in the sense of the de Morgan dual, except for atomic formulas $p^{\perp}$.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 246–264, 2009.

computational complexity classes. Among others, we can here recall the works by Asperti, Baillot, Lafont, Mairson, Roversi, Terui (see [1]-[16]). The intuitionistic fragments of a number of systems are shown to capture polynomial time computation within the computation-as-cut-elimination paradigm.

However, the underlying light logics do not behave well with respect to any choice of cut reduction strategies. Thus the original LLL has been proved to be only weakly polytime sound, where, in particular, we allow only a *lazy* cut-elimination procedure not reducing the so-called additive commutative cuts[8]. Asperti's Intuitionistic Light Affine Logic[1] works well as a calculus of intuitionistic proof-nets or as Terui's term calculus[15] (the latter enjoys polytime strong normalization), but due to unrestricted weakening cut elimination becomes non-deterministic within the classical version of Light Affine Logic.

The original syntax of LLL is too complicated, mainly because one has to deal with sequents which not just consist of formulas but also of 'blocks' of formulas. We circumvent the complications of 'blocks' by introducing a new modality ∇ which is exclusively in charge of 'additive blocks'. The most interesting feature of this purely multiplicative ∇ is the possibility of the second-order encodings of additive connectives. The resulting system (with the traditional syntax), called Easy-LLL, is still powerful to represent any deterministic polynomial time computations in purely logical terms. Unlike the original LLL and the classical version of light affine logic, Easy-LLL admits polynomial time strong normalization, namely, cut elimination terminates in a *unique way* in polytime by any choice of cut reduction strategies.

1.1 Light Linear Logic: Basics

LLL is dealing with the *multiplicative connectives*: $\otimes$ and its dual $\parr$, *additive connectives*: & and its dual $\oplus$, and *modalities*: ! and its dual ?, § and its dual $\S^{\perp}$.

LLL is designed to accommodate the following modalities properties[8]:

$$(!A \otimes !A) = !A,\ !A \vdash 1,\ !(A \& B) = (!A \otimes !B),\ \frac{A \vdash C}{!A \vdash !C},\ !A \vdash \S A,\ \frac{\Gamma \vdash C}{\S\Gamma \vdash \S C}. \quad (1)$$

As compared to traditional sequent calculi, the syntax of LLL is much more complicated. At the same level of formulas, it invokes also 'blocks' of formulas, such as a 'discharged formula' $[A]$ and a 'comma expression' $\langle A_1, A_2, .., A_\ell \rangle$. As a result, the application of the traditional cut-elimination argument for LLL sequent calculus runs into essential technical difficulties, even within the framework of proofnets.

The aim of the paper is to simplify LLL, resulting in a traditional sequent calculus, called Easy-LLL, that meets the fundamental complexity-theoretic constraints of the original LLL, and, on the other hand, enjoys polynomial-time strong normalization and the Church-Rosser property, and the strong representability of polytime computation.

Table 1. LLL from Girard[8] + "$\vdash !1$". § and $\S^{\perp}$ are duals of each other.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp};\, A}$(identity)	$\dfrac{\vdash \Gamma;\, A \qquad \vdash A^{\perp};\, \Theta}{\vdash \Gamma;\, \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma;\, A \quad \vdash \Theta;\, B}{\vdash \Gamma;\, \Theta;\, (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma;\, A;\, B}{\vdash \Gamma;\, (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma;\, \bot}$(false)
Additives:	
$\dfrac{\vdash \Gamma;\, A \quad \vdash \Gamma;\, B}{\vdash \Gamma;\, (A \& B)}$(with) $\quad \dfrac{}{\vdash \Gamma;\, \top}$(true)	$\dfrac{\vdash \Gamma;\, A}{\vdash \Gamma;\, (A \oplus B)}$(plus$_1$) $\quad \dfrac{\vdash \Gamma;\, B}{\vdash \Gamma;\, (A \oplus B)}$(plus$_2$)
Structure/Modalities:	
$\dfrac{\vdash \Gamma;\, A_i}{\vdash \Gamma;\, \langle A_1, .., A_i, .., A_\ell \rangle}$(**A**-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma;\, \langle A_1, .., A_\ell \rangle}$(**A**-weakening, $\ell \geq 1$)
$\dfrac{\vdash \Gamma;\, [A];\, [A]}{\vdash \Gamma;\, [A]}$(**M**-contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma;\, [A]}$(**M**-weakening) $\quad \dfrac{\vdash \Gamma;\, [A]}{\vdash \Gamma;\, ?A}$(?-intro)
$\dfrac{\vdash \langle D_1, .., D_\ell \rangle;\; C}{\vdash\; [D_1];\, ..;\, [D_\ell];\; !C}$(!-rule, $\ell \geq 0$)	$\dfrac{\vdash \langle D_1 \rangle;\, ..;\, \langle D_n \rangle;\; A_1;\, ..;\, A_m;\; C}{\vdash\; [D_1];\, ..;\, [D_n];\; \S^{\perp} A_1;\, ..;\, \S^{\perp} A_m;\; \S C}$ (§-rule, $n, m \geq 0$)
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma;\, A(p)}{\vdash \Gamma;\, \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma;\, A(B)}{\vdash \Gamma;\, \exists p A(p)}$(there-exists)

To simplify our presentation,[2] we enrich the original LLL[8] with "$\vdash !1$" (see Table 1). We omit the *exchange rules*, since we will deal with multisets of formulas and 'blocks' (following [8], we use ";" as a separation mark).

Remark 1. To return to the original LLL[8], it suffices to restrict $l \geq 1$ within '!-rule', and omit the '**A**-weakening' rule. The following example explains the necessity of '**A**-weakening' for the enriched LLL.

Example 1. When we allow "$\vdash !1$", we get an important "$!p \vdash !(p \& 1)$":

$$\frac{\vdash !1 \qquad \vdash ?1^{\perp};\, ?p^{\perp};\, !(p \& 1)}{\vdash ?p^{\perp};\, !(p \& 1)}\ \text{cut}$$

Any candidate for a cut-free proof must be like this:

$$\dfrac{\dfrac{\dfrac{\vdash\; p^{\perp};\, p}{\vdash\; \langle p^{\perp} \rangle;\, p} \qquad \dfrac{???}{\vdash\; \langle p^{\perp} \rangle;\, 1}}{\vdash\; \langle p^{\perp} \rangle;\, (p \& 1)}}{\dfrac{\vdash\; [p^{\perp}];\, !(p \& 1)}{\vdash ?p^{\perp};\, !(p \& 1)}}$$

Hence, to save cut-elimination for LLL+"$\vdash !1$", we need, at the least, '**A**-weakening' of the form: $\dfrac{\vdash\; 1}{\vdash\; \langle p^{\perp} \rangle;\, 1}$

[2] Girard[8]: "Typically § will have a tendency to lose its self-duality, promotion with empty context can be added to LLL with immediate simplifications."

1.2 Step 1: The Nabla-Version of LLL (LLL-∇)

To simplify Table 1 further, we take advantage of the fact that almost all blocks in there include only one formula. Namely, a block of the form $[A]$ is replaced by the formula $?A$ with practically no difference[8]. Following the '?-intro' rule in letter and in spirit, we intend to replace a block of the form $\langle A\rangle$ with the formula $\triangle A$, where $\triangle$ is a new modality.

Assume that ∇ is dual to $\triangle$. Then '**A**-dereliction' and '**A**-weakening' provide that $\nabla A \vdash A$ and $\nabla A \vdash 1$. To save cut-elimination for this pair $(\nabla, \triangle)$ at the minimum cost, we need a rule of the form

$$\frac{\nabla B \vdash C}{\nabla B \vdash \nabla C}, \quad \text{or a more general version:} \quad \frac{\nabla \Gamma \vdash C}{\nabla \Gamma \vdash \nabla C}$$

Bringing all together, we define a nabla-version of LLL by Table 2.

Table 2. LLL-∇: The nabla-version of LLL. ∇ and $\triangle$ are duals of each other.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp};\, A}$(identity)	$\dfrac{\vdash \Gamma;\, A \qquad \vdash A^{\perp};\, \Theta}{\vdash \Gamma;\, \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma;\, A \quad \vdash \Theta;\, B}{\vdash \Gamma;\, \Theta;\, (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma;\, A;\, B}{\vdash \Gamma;\, (A \parr B)}$(par) $\qquad \dfrac{\vdash \Gamma}{\vdash \Gamma;\, \perp}$(false)
Structure/Modalities:	
$\dfrac{\vdash \Gamma;\, A_i}{\vdash \Gamma;\, \langle A_1, .., A_i, .., A_\ell\rangle}$(**A**-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma;\, \langle A_1, .., A_\ell\rangle}$(**A**-weakening, $\ell \geq 1$)
$\dfrac{\vdash \triangle D_1;\, ..;\, \triangle D_n;\, C}{\vdash \triangle D_1;\, ..;\, \triangle D_n;\, \nabla C}$(∇-rule, $n \geq 0$)	$\dfrac{\vdash \Gamma;\, \langle A\rangle}{\vdash \Gamma;\, \triangle A}$(△-intro)
$\dfrac{\vdash \Gamma;\, ?A;\, ?A}{\vdash \Gamma;\, ?A}$(?-Contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma;\, ?A}$(?-Weakening)
$\dfrac{\vdash \langle D_1, .., D_\ell\rangle;\; C}{\vdash ?D_1;\, ..;\, ?D_\ell;\; !C}$(!-rule, $\ell \geq 0$)	$\dfrac{\vdash \triangle D_1;\, ..;\, \triangle D_n;\, A_1;\, ..;\, A_m;\, C}{\vdash ?D_1;\, ..;\, ?D_n;\, \S^{\perp} A_1;\, ..;\, \S^{\perp} A_m;\, \S C}$ (§-rule, $n, m \geq 0$)
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma;\, A(p)}{\vdash \Gamma;\, \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma;\, A(B)}{\vdash \Gamma;\, \exists p A(p)}$(there-exists)

Definition 1. *We do not include the additives in our Table 2, since we will define the additives as the following abbreviations, with providing the additive rules of Table 1 (see also Remark 6):*

$$(A \,\&\, B) = \exists p\,(p \otimes \nabla(p \multimap A) \otimes \nabla(p \multimap B)); \qquad \top = \exists p\, p;$$
$$(A \oplus B) = \forall p\,((\nabla(A \multimap p) \otimes \nabla(B \multimap p)) \multimap p); \;\; 0 = \forall p\, p.$$

Proposition 1. *It is clear that LLL-∇ is a conservative extension of LLL+ "$\vdash !1$".*

Proof. Replacing ∇A by $(A\&1)$ does not violate inference rules. ∎

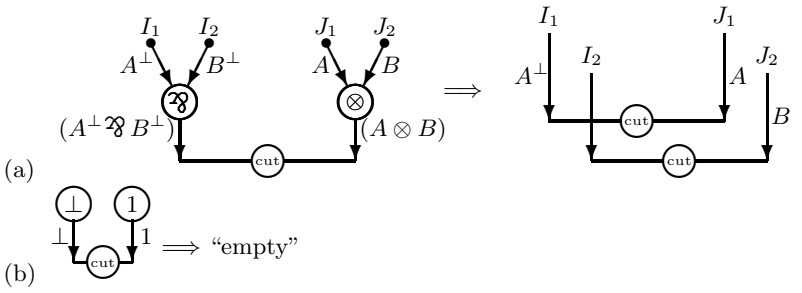


Fig. 1. The cut reduction rules for $(\otimes, \parr)$ and $(1, \bot)$

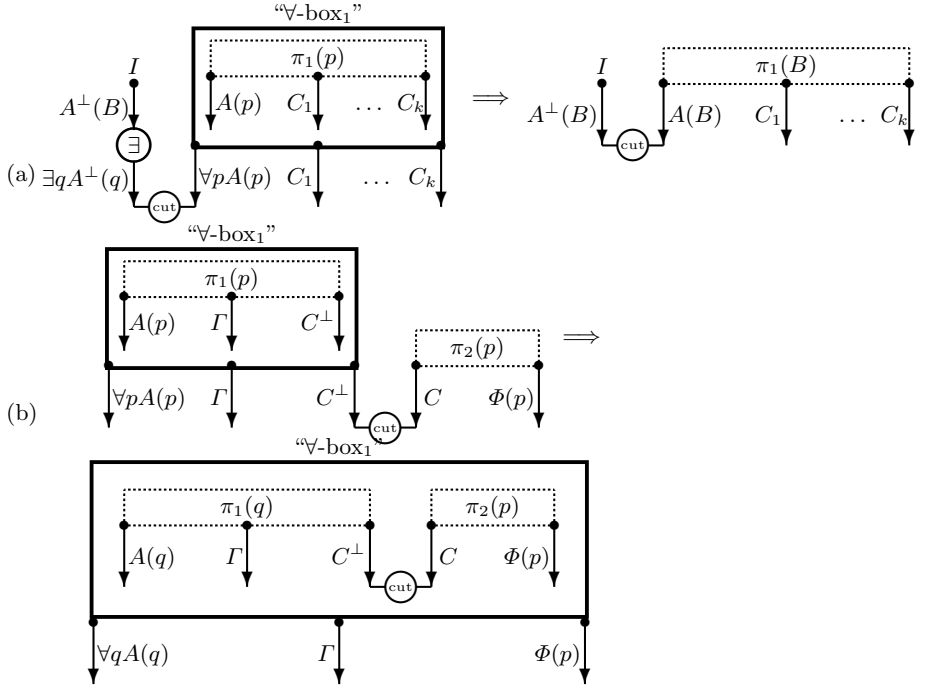


Fig. 2. (a) The $(\forall, \exists)$-cut reduction. (b) The (side,$\forall$)-cut reduction with renaming (q is fresh).

Phase Semantics for LLL-∇

Along the lines of [9], we introduce a fibred (stratified) phase semantics for LLL-∇. The basic idea is to interpret any formula of the new form ∇A as:

$$(\nabla A)^* = (A^* \cap J)^{\bot\bot},$$

where J is a fixed submonoid of a monoid M such that $J \subseteq \{e\}^{\bot\bot}$; here e denotes the neutral element of M.

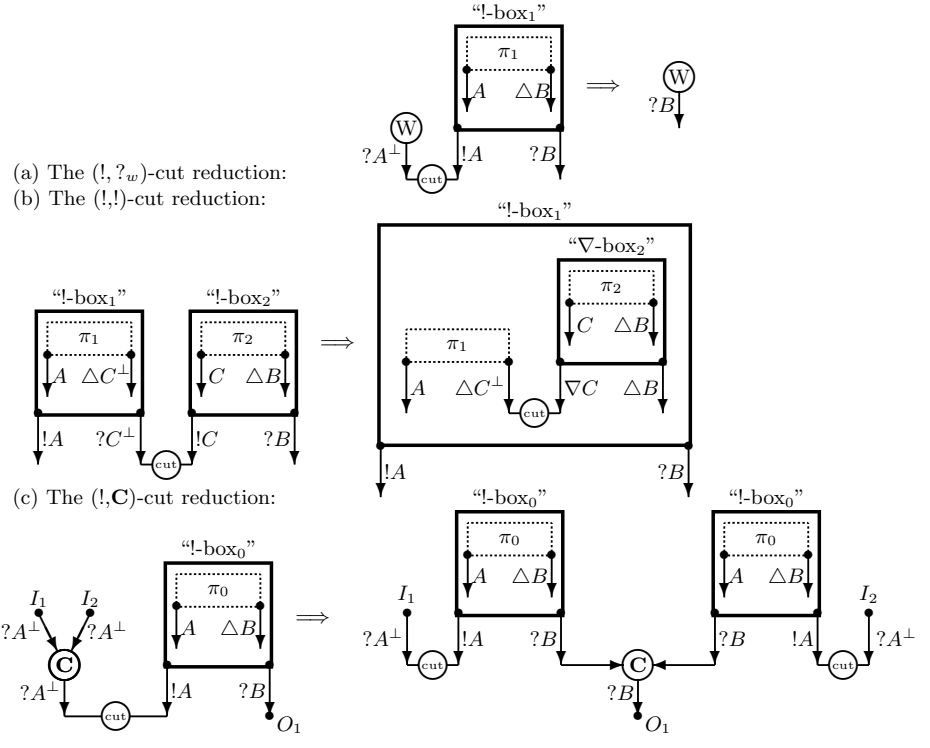


Fig. 3. The cut reduction rules for!

A *fibred phase space* is introduced as a sequence of phase spaces $\mathcal{M}_0$,$\mathcal{M}_1$,..., with providing a certain one-way interaction between adjacent $\mathcal{M}_{n+1}$ and $\mathcal{M}_n$ by means of functions h_n and f_n.

Definition 2. *A* phase space $(M, e, \perp)$ *is a commutative monoid M with a distinguished subset $\perp \subseteq M$; here e stands for the neutral element of M. For any subset $X \subseteq M$, define: $X^\perp = \{y \in M \mid \forall x \in X\,(xy \in \perp)\}$. A subset $X \subseteq M$ is* closed *iff $X^{\perp\perp} = X$. In particular, let $\mathbb{1}$ be the subset $\{e\}^{\perp\perp}$. The phase space induces a natural preorder on the underlying monoid compatible with monoid multiplication: $x \preceq y \Leftrightarrow x \in \{y\}^{\perp\perp}$.*

Definition 3. *A* homomorophism of phase spaces*, or simply a* phase homomorphism*, is a monoid homomorphism $h : M \to M'$ such that $h(\perp) \subseteq \perp'$.*

Given a partial mapping $f : M \to M'$, we say that f is bounded by h*, if $f(a) \preceq h(a)$ for every $a \in J$, where J is the domain of f.*

The mapping f has the intermediate value property *if for every a and b from J, there exists $c \in J$ such that $c \preceq a$ and $c \preceq b$, and $f(a)f(b) = f(c)$.* ■

Definition 4. *A* fibred phase space *is a sequence*

$$((M_n, e_n, \perp_n), h_n, J_n, f_n, L_n)_{n \geq 0},$$

where for each n,

(a) $(M_n, e_n, \perp_n)$ *is a phase space,* $h_n : M_{n+1} \to M_n$ *is a phase homomorphism,*
(b) J_n *is a submonoid of* $\mathbb{1}_n$*, and* $f_n : J_{n+1} \to M_n$ *is a mapping with the intermediate value property such that* f_n *is bounded by* h_n*, and, in addition,* $f_n(e_{n+1}) = e_n$*, and*
(c) L_n *is a submonoid of* $\mathbb{1}_n$ *such that every element* a *of* L_n *is a* weak idempotent*, i.e.,* $a \preceq_n a \cdot_n a$*.*

Definition 5. *Given a fibred phase space* $((M_n, e_n, \perp_n), h_n, J_n, f_n, L_n)_{n\geq 0}$*, for each formula* A *one associates a sequence of closed sets* A^{*_0}*,* A^{*_1}*,...,* A^{*_n}*,..., in the following inductive way, starting with any assignment of closed sets* p^{*_0}*,* p^{*_1}*,..., to propositional atoms* p*:*

$$\begin{array}{rlrl} 1^{*_n} &= \mathbb{1}_n, & \top^{*_n} &= M_n, \\ (A \otimes B)^{*_n} &= (A^{*_n} \cdot_n B^{*_n})^{\perp_n \perp_n}, & (A^{\perp})^{*_n} &= (A^{*_n})^{\perp_n}, \\ (A \& B)^{*_n} &= (A^{*_n} \cap B^{*_n}), & (\nabla A)^{*_n} &= (A^{*_n} \cap J_n)^{\perp_n \perp_n}, \\ (\S A)^{*_n} &= (h_n(A^{*_{n+1}}))^{\perp_n \perp_n}, & (!A)^{*_n} &= (f_n(A^{*_{n+1}} \cap J_{n+1}) \cap L_n)^{\perp_n \perp_n}. \end{array}$$

For a 'block' of the form $\langle A_1, A_2, .., A_\ell \rangle$*, define:*

$$\langle A_1, A_2, .., A_\ell \rangle^{*_n} = ((A_1^{*_n} \cup A_2^{*_n} \cup \cdots \cup A_\ell^{*_n})^{\perp_n} \cap J_n)^{\perp_n}.$$

The second-order case is considered as in [9]. A formula A *is* valid *in the fibred phase space, if for each* n*,* $e_n \in A^{*_n}$ *in any valuation* p^{*_0}*,* p^{*_1}*,..., to propositional atoms* p*.* ■

Theorem 1 (Strong Completeness)

(a) *If an* A *is provable in LLL-*∇*, then* A *is valid in any fibred phase space.*
(b) *If a formula* A *is valid in any fibred phase space, then* A *is provable in LLL-*∇ *without the cut rule.*

Corollary 1 (Cut-Elimination). *Theorem 1 provide a semantic proof for cut elimination in LLL-*∇*.*

Remark 2. Corollary 1 cannot give more than the pure existence of cut-free proofs. As for the traditional cut-reduction arguments where each of the cuts is reduced to a number of cuts by dealing only with a pair of inference rules, the problematic case caused by the full version of '!-rule' is given in Example 2.[3]

Example 2. The cut below cannot be eliminated by the traditional argument mentioned above:

$$\dfrac{\dfrac{\dfrac{\cdots}{\vdash \langle A^{\perp} \rangle;\, B}}{\vdash ?A^{\perp};\, !B} \qquad \dfrac{\dfrac{\cdots}{\vdash \langle B^{\perp}, C^{\perp} \rangle;\, D}}{\vdash ?B^{\perp};\, ?C^{\perp};\, !D}}{\vdash ?A^{\perp};\, ?C^{\perp};\, !D}\ (\text{cut})$$

[3] Girard[8]: "We shall define a *lazy* cut-elimination which terminates in polytime. The result of the procedure is cut-free only in certain cases, but this is enough for us."

Remark 3. Nevertheless, with certain modifications to handle **A***-blocks, we can define a syntactical strongly normalizing cut-elimination procedure for LLL-*∇*.*

Proof Sketch. We treat a block $\langle \Gamma \rangle$ as a 'generalized $\triangle$-formula', and replace '∇-rule' in Table 2 with:

$$\frac{\vdash \langle \Gamma_1 \rangle; \ldots; \langle \Gamma_n \rangle; C}{\vdash \langle \Gamma_1 \rangle; \ldots; \langle \Gamma_n \rangle; \nabla C} \ (\nabla\text{-rule}', \ n \geq 0) \tag{2}$$

To capture strange cuts between ∇B and the 'generalized $\triangle$-formula' that includes $B^{\perp}$, we add cut rules like this:

$$\frac{\vdash \langle \Gamma \rangle; \nabla B \qquad \vdash \langle B^{\perp}, \Phi \rangle; \Theta}{\vdash \langle \Gamma, \Phi \rangle; \Theta} \ (\text{cut}') \tag{3}$$

Accordingly, the cut in the problematic case of Example 2 will be reduced to:

$$\dfrac{\dfrac{\dfrac{\ldots}{\vdash \langle A^{\perp} \rangle; B}}{\vdash \langle A^{\perp} \rangle; \nabla B} \qquad \dfrac{\ldots}{\vdash \langle B^{\perp}, C^{\perp} \rangle; D}}{\dfrac{\vdash \langle A^{\perp}, C^{\perp} \rangle; D}{\vdash ?A^{\perp}; ?C^{\perp}; !D}} \ (\text{cut})$$

With additional cut reductions, we can provide termination but in *hyper-exponential* time. This huge bound is caused by the fact that we allow more than one conclusion of the form $?D_j$ within the "!-rule" (Cf. Tables 2 and 3). ■

Table 3. The Inference Rules of Easy-LLL. ∇ and $\triangle$ are duals of each other.

Identity / Negation:	
$\dfrac{}{\vdash A^{\perp}; A}$(identity)	$\dfrac{\vdash \Gamma; A \qquad \vdash A^{\perp}; \Theta}{\vdash \Gamma; \Theta}$(cut)
Multiplicatives:	
$\dfrac{\vdash \Gamma; A \quad \vdash \Theta; B}{\vdash \Gamma; \Theta; (A \otimes B)}$(times) $\quad \dfrac{}{\vdash 1}$(one)	$\dfrac{\vdash \Gamma; A; B}{\vdash \Gamma; (A \parr B)}$(par) $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma; \perp}$(false)
Modalities:	
$\dfrac{\vdash \Gamma; A}{\vdash \Gamma; \triangle A}$($\triangle$-dereliction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma; \triangle A}$($\triangle$-weakening)
$\dfrac{\vdash \triangle D_1; ..; \triangle D_n; C}{\vdash \triangle D_1; ..; \triangle D_n; \nabla C}$($\nabla$-rule, $n \geq 0$)	
$\dfrac{\vdash \Gamma; ?A; ?A}{\vdash \Gamma; ?A}$(?-Contraction)	$\dfrac{\vdash \Gamma}{\vdash \Gamma; ?A}$(?-Weakening)
$\dfrac{\vdash \triangle D_1; ..; \triangle D_\ell; C}{\vdash ?D_1; ..; ?D_\ell; !C}$ (!-rule, $\underline{\ell = 0, 1}$)	$\dfrac{\vdash \triangle D_1; ..; \triangle D_n; A_1; ..; A_m; C}{\vdash ?D_1; ..; ?D_n; \S^{\perp} A_1; ..; \S^{\perp} A_m; \S C}$ (§-rule, $n, m \geq 0$)
Second-Order Quantifiers:	
$\dfrac{\vdash \Gamma; A(p)}{\vdash \Gamma; \forall p A(p)}$(for-all, p is not free in Γ)	$\dfrac{\vdash \Gamma; A(B)}{\vdash \Gamma; \exists p A(p)}$(there-exists)

1.3 Step 2: The Easy Fragment of LLL (Easy-LLL)

Here we introduce a traditional syntactical fragment of LLL-∇, called Easy-LLL, which is of our main interest.

Notice that the syntax of Table 2 is almost perfect: only one rule, the '!-rule', invokes a block of formulas. To get rid of blocks of formulas, we will confine '!-rule' to $\ell \leq 1$ and replace a block $\langle D \rangle$ with the formula $\triangle D$. In Table 3 we represent the resulting Easy fragment of LLL, which deals with formulas only.

Remark 4. From the provability point of view, Easy-LLL is weaker than LLL-∇. One of the basic principles in LLL[8] is the *'exponential isomorphism'* (see (1)):

$$(!p \otimes !q) = !(p \& q).$$

One direction: $!(p\&q) \vdash (!p \otimes !q)$, is easily provable in Easy-LLL. As for other direction: $(!p \otimes !q) \vdash !(p\&q)$, it is provable in LLL-$\nabla$ (for the cost of its 'mixed' syntax with **A**-blocks), but it is not provable in Easy-LLL. It is unclear how to incorporate $(!p \otimes !q) \vdash !(p\&q)$ into a traditional sequent calculus formalism that admits cut elimination. E.g., by allowing $\ell = 2$ in the "!-rule" in Table 3, we 'jump' to *elementary functions* computable in *hyper-exponential time*.

On the other hand, what we actually need to represent polytime computation in [8,2] is a weaker form of the 'exponential isomorphism': $(!p \otimes !1) = !(p\&1)$, which is provable in Easy-LLL, as well. ■

Remark 5. We can show that Easy-LLL is sound and complete with respect to the *fibred phase semantics* obtained from Definition 4 by omitting the requirement that "f_n has the intermediate value property".

In Section 2 we show that Easy-LLL enjoys *strong normalization in polytime*, with providing a *unique cut-free form*. Namely, any proof is normalizable in s^d steps regardless of which sequence of cut reductions we take (here s is the size of the proof, and the degree d is determined only by the nesting depth of exponentials in the proof).

In Section 3 we show that Easy-LLL has the full expressive power of light logics, and captures exactly polytime computation.

2 Easy-LLL: Strong Normalization in Polytime

In this section we will prove polytime strong normalization for Easy-LLL.

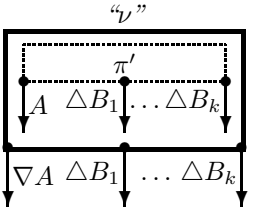
Definition 6. *Any proof within Easy-LLL can be represented as a proofnet π [8].*

In full accordance with the rules from Table 3, such a proofnet *π consists of nodes and boxes, some of them are connected by arrows, so that*

(a) *A node labelled by "axiom" has only two outgoing arrows labelled by formulas of the form A and $A^{\perp}$, resp. A node labelled by "cut" has only two incoming arrows labelled by formulas of the form A and $A^{\perp}$, resp.*

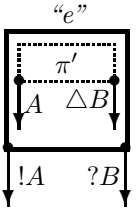
(b) *A node labelled by "$\otimes$" or "$\parr$" has exactly two incoming arrows, labelled by some A and B, and one outgoing arrow labelled by $(A \otimes B)$ or $(A \parr B)$, resp. A node labelled by "1" or "$\bot$" has only an outgoing arrow labelled by 1 or $\bot$.*

(c) *A node labelled by "$\triangle$" has one incoming arrow, labelled by an A, and one outgoing arrow labelled by $\triangle A$. A node labelled by "w" (representing $\triangle$-weakening) has one outgoing arrow labelled by a formula of the form $\triangle A$. A node labelled by a ∇-box of the form*

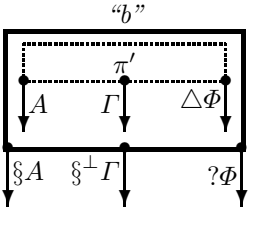


has $k+1$ outgoing arrows labelled by the corresponding conclusions of the ∇-box.

(d) *A node labelled by "$\mathbf{C}$" (representing ?-contraction) has two incoming arrows, labelled by one and the same $?A$, and one outgoing arrow labelled by the same $?A$. A node labelled by "W" (representing ?-weakening) has one outgoing arrow labelled by a formula of the form $?A$. A node labelled by a !-box of the form*

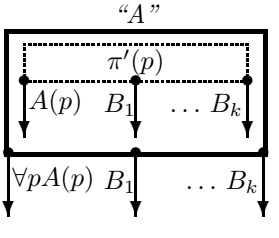


has two outgoing arrows labelled by $!A$ and $?B$. A node labelled by the §-box



where Γ and Φ are multisets of formulas, has outgoing arrows labelled by the conclusions of the §-box.

(e) *A node labelled by "$\exists$" has one incoming arrow, labelled by some $A(B)$, and one outgoing arrow labelled by $\exists p A(p)$. A node labelled by the $\forall$-box*



has $k+1$ outgoing arrows labelled by the conclusions of the $\forall$-box. Here $\pi'(p)$ is a proofnet with a marked variable p, and each of the conclusions $B_1, \ldots, B_k$ has no free occurrences of p. The propositional variable p is considered to be bound *within the $\forall$-box.*

(f) *An unlabelled node has only an incoming arrow labelled by an A, which is said to be a* conclusion *of π.* ■

Definition 7. *The* size *of a proofnet π is defined as the total number of all nodes and boxes contained inside π.*

In order to control the size of proofnets under cut reductions (see Lemma 1), for any box E, we introduce an adjusted size $w(E)$*, call it the* weight *of box E, as follows: $w(E) = s(\pi') + 2$, where $s(\pi')$ is the size of the proofnet π' inside E.*

For a given nesting depth l of !-boxes and §-boxes inside π (we call it 'level l'), we take a 'slice' $\pi^{(l)}$ on level l *as the collection of all nodes and boxes (together with arrows) on the depth l. The number of these nodes and boxes is called the* size of π' on level l *(each box is counted as a one node).*

Definition 8. *Following the rules in Table 3, we develop a natural set of cut reduction rules, which are collected in Figures 1, 3, 6, 7, and 2.*

Lemma 1. *We will list the basic peculiarities of our cut reduction rules:*

(a) *Each of the reductions does not mix levels.*

(b) *All reductions contract the size of a proofnet, except for (∇, ∇), $(!, !)$, $(!, §)$, $(§, §)$, $(!, C)$, and $(side, \forall)$.*

(c) *(∇, ∇) and $(side, \forall)$ preserve the size of a proofnet (and even the names of boxes), but put the cut at hand inside the corresponding expanded "box_1". The effect is that this cut will not directly contact with "box_1" in future.*

(d) *$(!, !)$, $(!, §)$ and $(§, §)$ merge "box_1" and "box_2" so that the weight of the new box (named as "box_1") does not exceed the sum of the weights of old boxes.*

(e) *$(!, C)$ moves a box E' from the position below a* **C***-node to the positions above the* **C***-node, with E' being doubled. A forest structure of* **C***-nodes stable under cut reductions is discussed in Definition 9 and Lemma 2.*

Definition 9. *We say that a* **C***-node v_2 is a* **C**-son *of a* **C***-node v_1, if these* **C***-nodes are connected by an alternating chain of !-boxes and cuts as shown in Figure 4. (The empty chain is allowed.)*

Lemma 2. *Let π be a proofnet constructed for a proof within Easy-LLL. Then the transitive closure of the relation "being a* **C***-son" from Definition 9 is acyclic, and* **C***-nodes form a forest with respect to this relation. Moreover, given a level l of π, the set of* **C***-nodes on level l remains intact by any of cut reductions on levels $\geq l$. If v_2 was a* **C**-son *of v_1 beforehand, then v_2 will be a* **C**-son *of v_2 after the cut reduction has been performed. So that the forest of* **C***-nodes formed according to "being a* **C***-son" can only grow up. Indeed, weakening cut reductions, such as $(\nabla, \triangle_w)$-cut reduction, can remove some trees from the forest.*

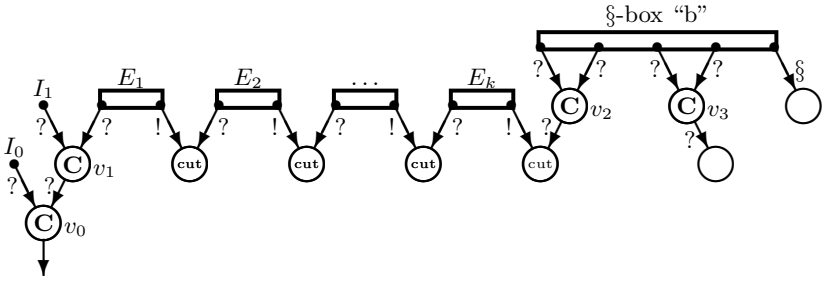


Fig. 4. Here v_1 is a **C**-*son* of v_0, and v_2 is a **C**-*son* of v_1. The nodes v_2 and v_3 are *leaves* in the forest formed by **C**-nodes w.r.t. "being a **C**-son".

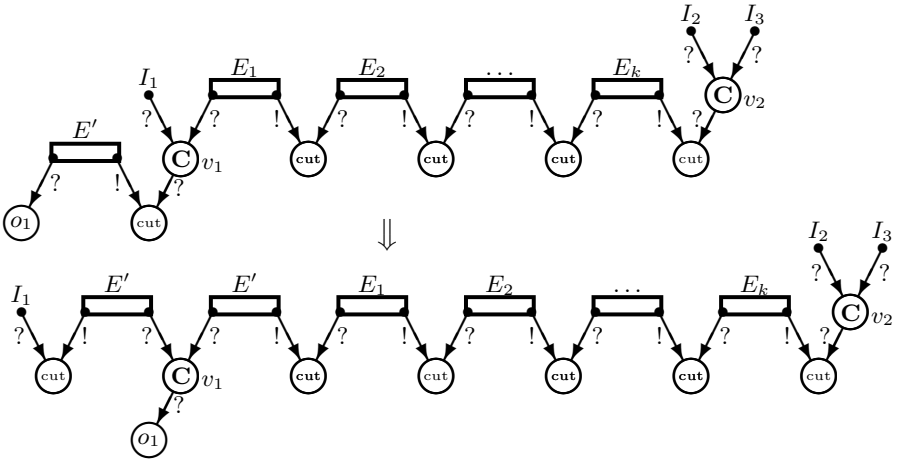


Fig. 5. An !-box E' moves through v_1, resulting in that a twin copy of E' appears on each of the branches of v_1.

Proof. The key case of $(!, C)$ reduction is shown in Figure 5.

The $(!,!)$, $(!,§)$, and $(§,§)$ reductions do not change the status of "being a **C**-son". Other reductions cannot destroy "being a **C**-son". Some reductions such as $(!,§)$ reduction, or say $(\otimes, ⅋)$ reduction, might have had the effect of seemingly 'merging' different branches in the trees of **C**-nodes by means of a §-box "b" like in Figure 4. But, according to Definition 9, which requires only !-boxes, the corresponding nodes v_2 and v_3 remain *leaves* in the forest of **C**-nodes. ■

Theorem 2. *Let π be a proofnet constructed for a proof within Easy-LLL.*

Then whatever sequence of cut reductions we take, it terminates in s^d steps, with the size of any reduct being bounded by s^d, where s is the size of π, and the degree d is determined only by the maximum nesting depth of !-boxes and §-boxes inside π.

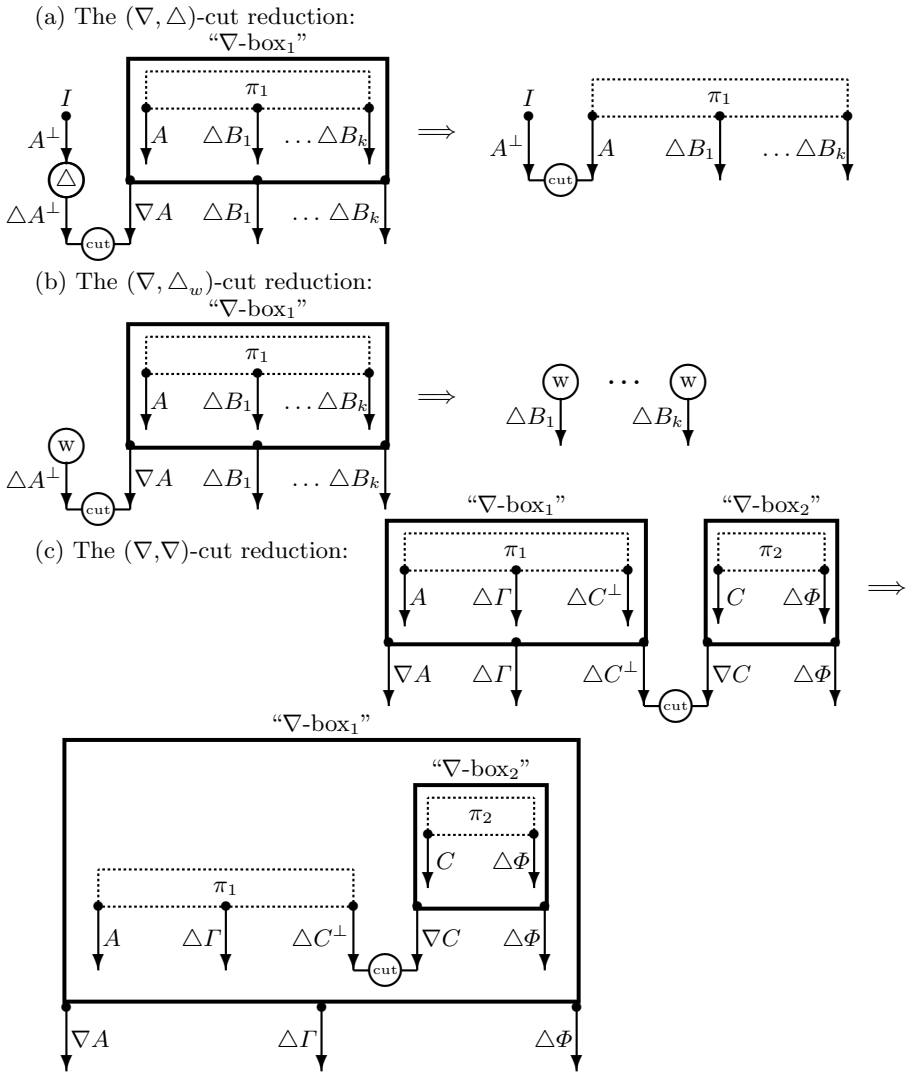


Fig. 6. The cut reduction rules for ∇

Proof Sketch. For each depth l, let s_l be the size of the 'slice' $\pi^{(l)}$ on level l; n_l be the number of ∇-boxes and $\forall$-boxes on level l; e_l be the number of !-boxes on level l; and C_l be the number of **C**-nodes that are leaves in the forest formed by **C**-nodes on level l according to Lemma 2.

Assume cut reductions α and β be applied in a row (α, then β), and α be applied on level l, and β be applied on level l' such that $l' < l$. It is clear that these two reductions can commute, and the number of reductions does not decrease. E.g., for β being a (!,**C**)-cut reduction, the resulting reductions are β, α, α.

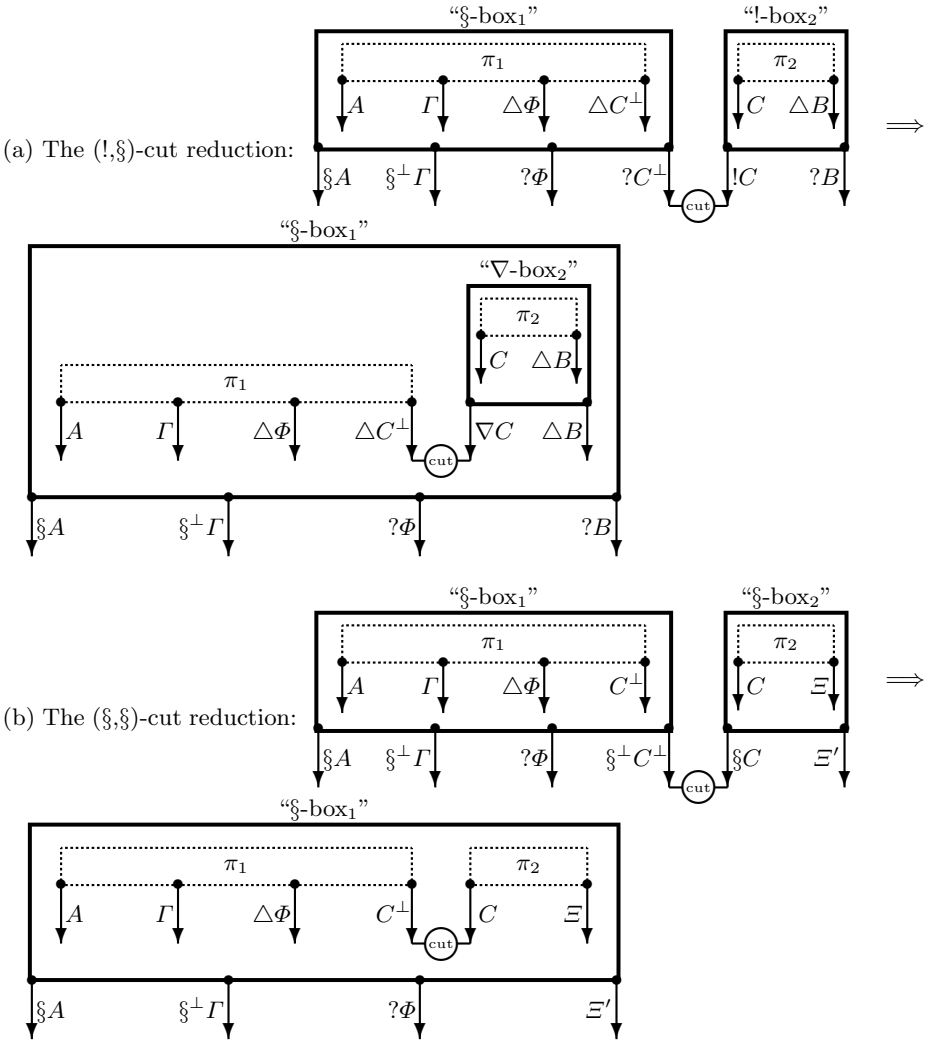


Fig. 7. The cut reduction rules for ! and §

Hence, the 'worst-case scenario' is, first, to reduce all cuts on level 0, then on level 1, then on level 2, and so on.

By examining non-size-contracting rules, we start with a rough estimate for the number of cut reductions on level 0.

According to Lemma 1, each of the !-boxes that is in cut-contact with a **C**-node from below (see Figure 5) will be multiplied and move up eventually beyond **C**-nodes that are leaves in the forest formed by **C**-nodes on level 0. The maximum number of the $(!,!)$ and $(!,C)$ reductions that could have been involved in is bounded by C_0^2, and the maximum number of the additional !-boxes produced

in the process is bounded by $2C_0 \cdot e_0$. Therefore, the size of $\pi^{(0)}$ in the process will be always bounded by $s_0 + 2C_0 e_0$.

Since a particular cut within (∇, ∇) and $(side, \forall)$ reductions cannot directly contact with the same "box$_1$" twice, the number of such reductions is bounded by $s_0 \cdot n_0$.

Bringing all together, the number of cut reductions on level 0 is bounded by $5s_0^2 \leq s_0^3$. In addition, Lemma 1 also provides that the size of $\pi^{(1)}$ will be kept in range of $6s_0^2 s_1^2 \leq s_0^3 s_1^2$.

For simplicity, let $s_l = s_0$, for all l.

By repeatedly applying the above procedure, we conclude that every sequence of cut reductions must terminate at most in $s_0^3 + s_0^{15} + s_0^{75} + \cdots$ steps, which yields a polynomial bound of the form s^d. ■

Theorem 3. *Our system of cut reduction rules is locally confluent.*

Proof. By examining all critical pairs. ■

Corollary 2. *Our system of cut reductions has the Church-Rosser property. In addition, every sequence of cut reductions terminates in polynomial time.*

Proof. This follows from Theorems 2 and 3. ■

3 The Expressive Power of Easy-LLL

Corollary 3. *Any polytime computable function is representable as a proof within Intuitionistic Easy-LLL.*

Proof. This easily follows from Girard-Asperti-Roversi's result [8,14,2], as well as from Murawski-Ong's interpretation of Bellantoni-Cook's safe recursion[13]. ■

Remark 6. In fact, Easy-LLL yields immediate simplifications. For instance, we can take advantage of ∇ in Definition 1 to provide "for free" the correct simulation of the additive boxes and related projections and conditionals within classical proofnets (cf. Girard[8]):

(1) The 'first projection', presented with 'plus$_1$', in a proof of the form (π_0, π_1, π_2 are cut-free):

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \Gamma;\, B} \qquad \dfrac{\pi_2}{\vdash \Gamma;\, B}}{\vdash \Gamma;\, (B \& B)} \qquad \dfrac{\dfrac{\pi_0}{\vdash \Phi;\, B^{\perp}}}{\vdash \Phi;\, (B^{\perp} \oplus B^{\perp})}\,\text{plus}_1}{\vdash \Gamma;\, \Phi}\,\text{cut}$$

is reduced to the first component of the pair: $\dfrac{\dfrac{\pi_1}{\vdash \Gamma;\, B} \quad \dfrac{\pi_0}{\vdash \Phi;\, B^{\perp}}}{\vdash \Gamma;\, \Phi}\,\text{cut}$

(2) The 'second projection', presented with 'plus$_2$', in a proof of the form:

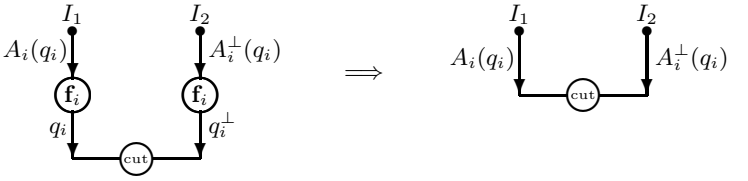


Fig. 8. The cut reduction in the case of fixed-point rules

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \Gamma;\, B} \qquad \dfrac{\pi_2}{\vdash \Gamma;\, B}}{\vdash \Gamma;\, (B\&B)} \qquad \dfrac{\dfrac{\pi_0}{\vdash \Phi;\, B^\perp}}{\vdash \Phi;\, (B^\perp \oplus B^\perp)}\,\mathrm{plus}_2}{\vdash \Gamma;\, \Phi}\,\mathrm{cut}$$

is reduced to the second component of the pair: $\dfrac{\dfrac{\pi_2}{\vdash \Gamma;\, B} \qquad \dfrac{\pi_0}{\vdash \Phi;\, B^\perp}}{\vdash \Gamma;\, \Phi}\,\mathrm{cut}$

Notice that ∇A itself can be conceived of as a 'projection', since $\nabla\nabla A = \nabla A$.

Remark 7. Theorem 2 gives the exact upper bound for the expressive power of Easy-LLL and its generalizations.

Corollary 4. *Let f be a function representable as a proof in Easy-LLL enriched with a number of fixed-point rules:*

$$\frac{\vdash \Gamma;\, A_i(q_i)}{\vdash \Gamma;\, q_i} \quad \textit{and} \quad \frac{\vdash \Gamma;\, A_i^\perp(q_i)}{\vdash \Gamma;\, q_i^\perp} \quad i=1,2,\ldots$$

where each of these $A_i(q_i)$ is a formula with a distinguished propositional variable q_i, the fixed-point of A_i. Then f is computed in polynomial time. The degree of the polynomial is determined by the nesting depth of !*-boxes and* §*-boxes within a representation of f.*

Proof. Theorem 2 still remains valid, when we invoke the cut reductions for fixed-points in Figure 8. ■

4 Concluding Remarks

The 'nabla' modality has appeared here as a result of consecutive attempts to simplify the original 'mixed' syntax of Girard's *light linear logic* (LLL) in accordance with the basic complexity-theoretic constraints of LLL[8].

First, we extend the original LLL by adding two rules: one that introduces $\vdash !1$, and '**A**-weakening' that allows weakening on additive blocks (see Table 1).

The second step introduces LLL-∇ (see Table 2) by:

– Substituting $?A$ for every occurrence of a block of the form $[A]$.
– Substituting $\triangle A$ for every occurrence of a block of the form $\langle A\rangle$, where $\triangle$ is dual to ∇. Indeed, it is observed that $\nabla A \vdash A$ and $\nabla A \vdash 1$ are derivable in LLL-∇.
– Introducing '∇-rule', which is a non functorial promotion that preserves cut elimination.

Remark 8. From a purely logical point of view, we obtain a logical system with three modalities where everything is under control:

(i) The § modality is a multi-functorial modality that raises a level: $A;\ B \vdash C$ yields $\S A;\ \S B \vdash \S C$.

(ii) A functorial modality ! is applied to formulas on which Contraction and Weakening can apply but on the next level.

The main observations about the morphisms it defines are $!A \vdash (!A \otimes !A)$, and $!A \vdash 1$, and $!A \vdash \S A$. See also Remark 4 about the *'exponential isomorphism'*.

(iii) The ∇ modality controls Weakening but on the current level.

In fact, our ∇A can be conceived of as a *purely multiplicative* connective that is served as the *greatest lower bound* of the *additive* $(A\&1)$. In particular, from the point of [6], ∇A can be interpreted as $!_1 A$. The most interesting feature of this purely multiplicative ∇ is the possibility of defining additive connectives without the need of unrestricted weakening (see Definition 1). In fact, in presence of restricted weakening, as in linear logic, the additive connectives can be encoded using second order quantification, but these encodings crucially need the exponential modalities:

$$(A \,\&\, B) = \exists p\,(p \otimes !(p \multimap A) \otimes !(p \multimap B)) \tag{4}$$

Such modalities enjoy weaker properties in LLL, and thus the standard encodings available in linear logic do not work for light logics (not even for *elementary light logic*). To express additives, light affine logic invokes unrestricted Weakening[2]. In presence of unrestricted weakening, as in affine logic, the additive connectives can be encoded as:

$$(A \,\&\, B) = \exists p\,(p \otimes (p \multimap A) \otimes (p \multimap B)) \tag{5}$$

But the cost of adding unrestricted weakening is the nondeterminism of cut-elimination in Classical Light Affine Logic:

Example 3. (*after Lafont*)
A proof of the form

$$\dfrac{\dfrac{\dfrac{\pi_1}{\vdash \Gamma}}{\vdash \Gamma;\, C^{\perp}}\,\mathrm{W} \qquad \dfrac{\dfrac{\pi_2}{\vdash \Phi}}{\vdash \Phi;\, C}\,\mathrm{W}}{\vdash \Gamma;\, \Phi}\,\mathrm{cut} \tag{6}$$

can be reduced to $\dfrac{\dfrac{\pi_1}{\vdash \Gamma}}{\vdash \Gamma;\, \Phi}$

But the same proof (6) can be also reduced to $\dfrac{\dfrac{\pi_2}{\vdash \Phi}}{\vdash \Gamma;\, \Phi}$

Example 3 destroys the Church-Rosser property for classical affine logics. This is the reason why, in the case of light affine logic, "in order to maintain the proofs-as-programs paradigm, we are compelled to stick to the intuitionistic version, in contrast to the case of Light Linear Logic, for which both classical and intuitionistic formulations are possible." (Terui[16])

It turns out that ∇ is actually all that we need to express additive connectives by the second-order encodings. See also Remark 6, which emphasizes the deterministic behaviour of cut elimination even for classical Easy-LLL.

Additive connectives are important because they can be used to obtain very natural representations of programming constructs such as conditionals. ■

The last step in clarifying the rules of the system is about Easy-LLL (see Table 3), obtained from LLL-∇ by:

- Restricting '!-rule' to the form in which $!C$ may depend on at most a single assumption of the form $\langle D \rangle$.
- Substituting $\triangle D$ for such a $\langle D \rangle$ in the just obtained '!-rule'.

We have proved that Easy-LLL, a traditional syntactical fragment of LLL, captures deterministic polytime computation in an 'ideal' way:

(a) The intuitionistic fragment of Easy-LLL is powerful enough to represent all polytime functions.
(b) Easy-LLL enjoys polytime strong normalization, which is within the paradigm: 'cut-elimination as a polytime computation'.
(c) Easy-LLL cut reductions enjoy the Church-Rosser property, which is within the paradigm: 'cut-elimination as a deterministic computation'.

Our proof of polytime strong normalization for Easy-LLL is based on the natural cut reductions directly produced from the rules of Easy-LLL.

The crucial non-trivial step of the proof is that we take advantage of a restricted form of '!-rule' in Table 3 to reveal a much deeper underlying forest structure of **C**-nodes, which is stable under reductions (see Definition 9 and Lemma 2). Having detected such a stable tree-like skeleton of **C**-nodes (which represent the most problematic ?-contraction rule) provides a transparent and clean construction for polynomial bounds in Theorem 2.

From the point of view of the complexity of applications, we believe that Easy-LLL provides a reasonable balance between its expressive power (to capture deterministic polytime computation) and its polytime strong normalization (to guarantee polytime upper bounds):

(a) Though the syntax of Easy-LLL is traditional, Easy-LLL integrates all basic principles of Girard's light linear logic.
(b) Easy-LLL has a convincing phase semantics.
(c) Easy-LLL enjoys a transparent polytime strong normalization.
(d) It is shown that even additive-free fragment of light linear logic is capable of expressing polytime computation [11], but their proof is extremely complicated.

Easy-LLL is flexible enough to incorporate basic constructs, such as additives/conditionals, to simulate polytime computation in a natural way (cf. [13,15,16,11]).

In closing, allowing $\ell = 2$ within '!-rule' in Table 3 yields *elementary linear logic*[8]. By the same argument we can obtain a transparent proof of strong normalization in elementary time for elementary linear logic[8].

Acknowledgments

I owe special thanks to the referees for their insightful comments and fruitful recommendations, which have allowed to improve the exposition of this paper.

References

1. Asperti, A.: Light affine logic. In: Proceedings of LICS 1998 (1998)
2. Asperti, A., Roversi, L.: Intuitionistic Light Affine Logic. ACM Transactions on Computational Logic (TOCL) 3(1), 137–175 (2002)
3. Atassi, V., Baillot, P., Terui, K.: Verification of Ptime reducibility for system F terms via Dual Light Affine Logic. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 150–166. Springer, Heidelberg (2006)
4. Baillot, P., Terui, K.: Light types for polynomial time computation in lambda-calculus. In: Proceedings of LICS 2004, pp. 266–275 (2004)
5. Girard, J.-Y.: Linear logic. Theoretical Computer Science 50(1), 1–102 (1987)
6. Girard, J.-Y., Scedrov, A., Scott, P.: Bounded linear logic: A modular approach to polynomial time computability. Theoretical Computer Science 97, 1–66 (1992)
7. Girard, J.-Y.: Linear logic: its syntax and semantics. In: Girard, J.-Y., Lafont, Y., Regnier, L. (eds.) Advances in Linear Logic, London Mathematical Society Lecture Notes, vol. 222, pp. 1–42. Cambridge University Press, Cambridge (1995)
8. Girard, J.-Y.: Light linear logic. Information and Computation 143(2), 175–204 (1998)
9. Kanovich, M., Okada, M., Scedrov, A.: Phase semantics for light linear logic. Theoretical Computer Science 294(3), 525–549 (2003)
10. Lafont, Y.: Soft Linear Logic and Polynomial Time. Theoretical Computer Science (special issue on Implicit Computational Complexity) 318, 163–180 (2004)
11. Mairson, H., Terui, K.: On the Computational Complexity of Cut-Elimination in Linear Logic. In: Blundo, C., Laneve, C. (eds.) ICTCS 2003. LNCS, vol. 2841, pp. 23–36. Springer, Heidelberg (2003)
12. Mazza, D.: Linear Logic and Polynomial Time. Mathematical Structures in Computer Science 16(6), 947–988 (2006)
13. Murawski, A.S., Ong, C.-H.L.: On an interpretation of safe recursion in light affine logic. Theoretical Computer Science 318(1/2), 197–223 (2004)
14. Roversi, L.: A P-time completeness proof for light logics. In: Flum, J., Rodríguez-Artalejo, M. (eds.) CSL 1999. LNCS, vol. 1683, pp. 469–483. Springer, Heidelberg (1999)
15. Terui, K.: Light affine lambda calculus and polytime strong normalization. In: Proceedings of LICS 2001, pp. 209–220 (2001)
16. Terui, K.: Light logic and polynomial time computation, PhD thesis, Keio University (2002)

Fuzzy Description Logic Reasoning Using a Fixpoint Algorithm*

Uwe Keller[1] and Stijn Heymans[2]

[1] Semantic Technology Institute (STI) Innsbruck, University of Innsbruck, Austria
uwe.keller@sti2.at
[2] Knowledge-based Systems Group, Institute of Information Systems, Vienna University of Technology, Austria
heymans@kr.tuwien.ac.at

Abstract. We present $\mathbf{FixIt}(\mathbb{ALC})$, a novel procedure for deciding knowledge base (KB) satisfiability in the Fuzzy Description Logic (FDL) $\mathbb{ALC}$. $\mathbf{FixIt}(\mathbb{ALC})$ does not search for tree-structured models as in tableau-based proof procedures, but embodies a (greatest) fixpoint-computation of canonical models that are not necessarily tree-structured, based on a type-elimination process. Soundness, completeness and termination are proven and the runtime and space complexity are discussed. We give a precise characterization of the worst-case complexity of deciding KB satisfiability (as well as related terminological and assertional reasoning tasks) in $\mathbb{ALC}$ in the general case and show that our method yields a worst-case optimal decision procedure (under reasonable assumptions). To the best of our knowledge it is the first *fixpoint-based* decision procedure for FDLs, hence introducing a new class of inference procedures into FDL reasoning.

1 Introduction

Description Logics (DLs) [1] are a popular family of formally well-founded and decidable knowledge representation languages. DLs have a wide range of applications, e.g., they form the basis for Semantic Web (SW) ontology languages used such as OWL [11]. Fuzzy Description Logics (FDLs) [17] extend DLs to represent *vague* concepts and relations, and as such are very well suited to cover for representing and reasoning with uncertainty, a requirement that naturally arises in many practical applications of knowledge-based systems, in particular the SW; FDLs for instance fit very well to the problem of multimedia information retrieval [9]. Another feature that makes FDLs specifically interesting for the SW is a basic form of para-consistency, i.e. a statement and its negation are possible to hold at the same time (to a certain extent). This allows knowledge providers on the SW to disagree on the basic properties of data objects and their interrelation without causing the (uninformative) explosion of the deductive closure as in classical DLs.

So far, reasoning in Fuzzy DLs is mainly based on tableau-methods (e.g.[3,7,15,16,17,20]). Further, [18] demonstrates how to use inference procedures for

* This work has been partially funded by the European Commission under the LarKC project (FP7 - 215535). Stijn Heymans is supported by the Austrian Science Fund (FWF) under projects P20305 and P20840.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 265–279, 2009.

classical DLs to perform reasoning in (some) FDLs. Still, reasoning in FDLs is at least as hard as reasoning in classical (crisp) DLs. Even in DLs of modest expressivity (e.g. $\mathbb{ALC}$ [17,18,16] the fuzzy variant of the DL $\mathcal{ALC}$ [14]) the worst-case complexity of reasoning is significant (cf. Section 3) even in restricted cases [17]. Therefore, it is clear that there can not be a *single* inference method that works well on *all* problems.

Consequently, our goal is to enrich the range of available methods for reasoning with FDLs with a fundamentally different approach. In practical applications of DLs (and hence FDLs) a particularly important feature for representing domain models is the support of so-called *general terminologies* (see e.g. [16]), i.e., the possibility to capture (potentially recursive) interdependencies between complex concepts in a domain model. However, besides the tableau-based methods for DLs (e.g [16,7,20,3]) there are at present no other FDL inference methods which can deal with general terminologies. We want to provide an alternative to tableau-based methods that can deal with general terminologies.

The main contributions of the paper are as follows:

- We present a novel procedure $\mathbf{FixIt}(\mathbb{ALC})$ (cf. Section 4.2) for deciding knowledge base (KB) satisfiability in the FDL $\mathbb{ALC}$ (cf. Section 2).
- We clarify the worst-case complexity of the reasoning task addressed by our algorithm and show formally that the problem is EXPTIME-complete. From this result, we can further establish EXPTIME-completeness for a range of related terminological and assertional reasoning tasks (cf. Section 3).
- We formally prove soundness, completeness and termination of the algorithm (cf. Section 4.2) and show that the runtime behavior of the proposed algorithm is worst-case optimal (cf. Section 4.3).
- $\mathbf{FixIt}(\mathbb{ALC})$ generalizes a type-elimination-based decision procedure [12] for the (classical) modal logic $\mathbf{K}$ (i.e. $\mathcal{KBDD}$ [10]) to the FDL $\mathbb{ALC}$. Additionally we integrate (fuzzy) ABoxes and general TBoxes which are not dealt with in $\mathcal{KBDD}$.
- To the best of our knowledge it is the first *fixpoint-based* decision procedure that has been proposed for FDL introducing a *new class of inference procedures* into FDL reasoning.
- Besides the tableau-based methods in [16,7,20,3], it is the only approach to integrate general terminologies in FDL reasoning and the first non-tableau-based one that we are aware of. General terminologies are handled in a fundamentally different way than in standard tableau-based method such as [16,7].

Our method is interesting especially regarding the last aspect since the handling of general terminologies in standard tableau-based methods (e.g. [16,7]) is a *major* source of non-determinism (cf. Section 5) and thus computational inefficiency. In our case no non-deterministic choice is introduced by terminologies.

2 Preliminaries

We introduce $\mathbb{ALC}$ [17], the fuzzy variant of the Description Logic $\mathcal{ALC}$ [14] (the latter can be seen as a syntactic variant of the multi-modal logic $\mathbf{K}_{(\mathbf{m})}$ [13]). $\mathbb{ALC}$ provides the starting point for more expressive FDLs [19] that have been proposed to fuzzify major fragments of OWL [11].

Syntax. Concept expressions are constructed from a signature $\Sigma = (\mathbf{C}, \mathbf{R}, \mathbf{I})$ with concept names $\mathbf{C}$, role names $\mathbf{R}$, and individual names $\mathbf{I}$. The set of concept expressions $\mathcal{C}(\Sigma)$ over Σ is defined as the smallest set of expressions that contains $\mathbf{C}$, $\top$ and is closed under the application of the concept constructors $C \sqcap D$ (intersection), $C \sqcup D$ (union), $\neg C$ (complement), and $\forall R.C$ (universal role restriction) for $R \in \mathbf{R}$ and $C, D \in \mathcal{C}(\Sigma)$. We allow expressions $\exists R.C$ for $C \in \mathcal{C}(\Sigma), R \in \mathbf{R}$ and $\bot$ and treat them as shortcuts for $\neg\forall R.\neg C$ and $\neg\top$ respectively. A TBox axiom (or general concept inclusion axiom (GCI)) is an expression of the form $C \sqsubseteq D$ s.t. $C, D \in \mathcal{C}(\Sigma)$. A terminology (or TBox) $\mathcal{T}$ is a finite set of TBox axioms. Syntactically, the vagueness of descriptions becomes explicit only when describing specific instances and their interrelations: a (fuzzy) ABox axiom is either a $\langle i : C \bowtie d\rangle$ or a $\langle R(i, i') \geq d\rangle$ s.t. $i, i' \in \mathbf{I}$, $d \in [0, 1]$, and $\bowtie \in \{\leq, \geq, =\}$. An ABox $\mathcal{A}$ is a finite set of ABox axioms. Finally, a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. Let $Ind_{\mathcal{A}} \subseteq \mathbf{I}$ denote the individual names that occur in $\mathcal{A}$. We denote the set of all concept expressions that occur as subexpressions in $\mathcal{K}$ by $\mathsf{sub}(\mathcal{K})$.

Semantics. Semantically, vagueness is reflected in the use of fuzzy sets and relations when interpreting concepts and roles: an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$ called the domain, and a function $\cdot^{\mathcal{I}}$ which maps each concept name $C \in \mathbf{C}$ to a fuzzy set $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$, each role name $R \in \mathbf{R}$ to a fuzzy relation $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$ and each individual name $i \in \mathbf{I}$ to an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is extended to arbitrary concept expressions $C \in \mathcal{C}(\Sigma)$ as follows: 1. $(C \sqcap D)^{\mathcal{I}}(o) = min(C^{\mathcal{I}}(o), D^{\mathcal{I}}(o))$ 2. $(C \sqcup D)^{\mathcal{I}}(o) = max(C^{\mathcal{I}}(o), D^{\mathcal{I}}(o))$ 3. $(\neg C)^{\mathcal{I}}(o) = 1 - C^{\mathcal{I}}(o)$ 4. $(\forall R.C)^{\mathcal{I}}(o) = inf_{o' \in \Delta^{\mathcal{I}}}\{max(1 - R^{\mathcal{I}}(o, o'), C^{\mathcal{I}}(o'))\}$ 5. $\top^{\mathcal{I}}(o) = 1$ for all $o \in \Delta^{\mathcal{I}}, C, D \in \mathcal{C}(\Sigma), R \in \mathbf{R}$. Note that, in contrast to classical DLs, it does not hold that $(C \sqcup \neg C)^{\mathcal{I}} = \top^{\mathcal{I}}$ for all interpretations $\mathcal{I}$, hence the need to add $\top$ (or $\bot$) to the language explicitly.

An interpretation $\mathcal{I}$ satisfies a TBox axiom $\alpha = C \sqsubseteq D$ iff for all $o \in \Delta^{\mathcal{I}}$ it holds that $C^{\mathcal{I}}(o) \leq D^{\mathcal{I}}(o)$, i.e. C is a fuzzy subset of D. $\mathcal{I}$ satisfies an ABox axiom $\alpha = \langle i : C \bowtie d\rangle$ iff $C^{\mathcal{I}}(i^{\mathcal{I}}) \bowtie d$. $\mathcal{I}$ satisfies an ABox axiom $\alpha = \langle R(i, i') \geq d\rangle$ iff $R^{\mathcal{I}}(i^{\mathcal{I}}, i'^{\mathcal{I}}) \geq d$. In all these cases, we write $\mathcal{I} \models \alpha$. $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ (or is a model of $\mathcal{T}$) iff $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{T}$. $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ (or is a model of $\mathcal{A}$) iff $\mathcal{I} \models \alpha$ for all $\alpha \in \mathcal{A}$. Finally, $\mathcal{I}$ satisfies a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (or is a model of $\mathcal{K}$) iff $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$.

Reasoning in $\mathbb{ALC}$. Given a fuzzy KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, fuzzy ABox axioms or GCIs α and concept expressions $C, D \in \mathcal{C}(\Sigma)$, we can analyze particular semantic characteristics and interdependencies: We say that $\mathcal{K}$ is *satisfiable* (or consistent) iff there is a model $\mathcal{I}$ for $\mathcal{K}$. $\mathcal{K}$ *entails* α (denoted as $\mathcal{K} \models \alpha$) iff all models $\mathcal{I}$ of $\mathcal{K}$ satisfy α. Concept C is subsumed by concept D (wrt. a KB $\mathcal{K}$) iff $\mathcal{K} \models C \sqsubseteq D$. Two concepts C and D are called *equivalent* (wrt. a KB $\mathcal{K}$) iff for any model $\mathcal{I}$ of $\mathcal{K}$ it holds that $C^{\mathcal{I}}(o) = D^{\mathcal{I}}(o)$ for all $o \in \Delta^{\mathcal{I}}$. Two concepts C and D are called *disjoint* (wrt. a KB $\mathcal{K}$) iff for any model $\mathcal{I}$ of $\mathcal{K}$ it holds that there does not exists an $o \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(o) > 0$ and $D^{\mathcal{I}}(o) > 0$. A concept C is called satisfiable (wrt. a KB $\mathcal{K}$) iff there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}(o) > 0$ for some $o \in \Delta^{\mathcal{I}}$. Further, one might want to compute the truth

value bounds for a given ABox assertion α wrt. $\mathcal{K}$ to determine the possibility interval that is enforced for α by the background knowledge in $\mathcal{K}$: The *greatest lower bound* of α wrt. $\mathcal{K}$ is defined as $glb(\alpha, \mathcal{K}) := sup\{d \,|\, \mathcal{K} \models \langle \alpha \geq d \rangle\}$ and the *least upper bound* of α wrt. $\mathcal{K}$ is defined as $lub(\alpha, \mathcal{K}) := inf\{d \,|\, \mathcal{K} \models \langle \alpha \leq d \rangle\}$ (where $sup\, \emptyset = 0$ and $inf\, \emptyset = 1$). Computing $glb(\alpha, \mathcal{K})$ and $lub(\alpha, \mathcal{K})$ is usually called the *best truth value bounds* (BTVB) problem.

One of the most fundamental reasoning problems is to determine whether a given fuzzy KB $\mathcal{K}$ is satisfiable. A lot of other reasoning tasks (e.g., checking for concept satisfiability wrt. a TBox or the BTVB problem) can be reduced to KB satisfiability checking [17] and therefore solved by a respective decision procedure. For this reason, we consider KB satisfiability as the reasoning problem to be solved.

3 Complexity of Reasoning with Knowledge Bases

Deciding the satisfiability of KBs in $\mathbb{ALC}$ where the TBox $\mathcal{T}$ is restricted to axioms of the form $A \sqsubseteq C$ or $A \equiv C$ (for concept names $A \in \mathbf{C}$ and concept expressions $C \in \mathcal{C}(\Sigma)$) such that any concept name A occurs at most once on the left-hand side and the TBox does not contain any cyclic dependencies between concept names is known to be a PSPACE-complete problem [17]. For the general case of unrestricted terminologies (allowing arbitrary GCIs $C \sqsubseteq D$), we are not aware of any worst-case complexity characterization. We show that determining the satisfiability of a KB in the general case is EXPTIME-complete (which corresponds to the situation in the classical variant $\mathcal{ALC}$). Detailed proofs are omitted here but can be found in [6].

EXPTIME-*Hardness.* We show EXPTIME-hardness by a polynomial-time (many-one) reduction of concept satisfiability wrt. general terminologies in the classical DL $\mathcal{ALC}$ which is known to be an EXPTIME-hard problem [1, Chapter 3]. We define the necessary reduction function as follows:

Definition 1 (Reduction). *Let* T *denote a finite set of GCIs and* $C \in \mathcal{C}(\Sigma)$ *be any concept expression. Then we define the reduction* $\pi(\mathrm{T}, C)$ *of the terminology* T *and the concept expression* C *as* $\pi(\mathrm{T}, C) := \mathcal{K}$ *where* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *is the* $\mathbb{ALC}$ *knowledge base consisting of the TBox* $\mathcal{T} := \mathrm{T} \cup \mathcal{T}^*$ *with* $\mathcal{T}^* := \{\top \sqsubseteq \neg D \sqcup D \,|\, D \in \mathsf{sub}(\mathrm{T} \cup \{C\})\}$ *and the ABox* $\mathcal{A} := \{\langle i : C \geq 1 \rangle\}$ *for some new individual name* i*.*

The intention of the additional TBox components $\mathcal{T}^*$ ensures that any model of $\mathcal{T}$ (and hence $\pi(\mathrm{T}, C)$) assigns only possibility degrees in $\{0, 1\}$ to any concept (sub-) expression occurring somewhere in T or C. In particular, any such model always assigns classical truth values to any concept name $A \in \mathbf{C}$ and any concept expression that is constructed from a role $R \in \mathbf{R}$ (but not necessarily to the roles $R \in \mathbf{R}$ themselves). Since for possibility degrees in $\{0, 1\}$ the (fuzzy) semantics of concept constructors in $\mathbb{ALC}$ coincides with the classical semantics in $\mathcal{ALC}$, we know that a fuzzy interpretation that satisfies $\mathcal{T}$ can be modified (easily) into a classical (i.e. crisp) model of T. By the same line of argumentation, $\mathcal{A}$ ensures that in any model of $\pi(\mathrm{T}, C)$ (and hence $\mathcal{T}$) the input concept C is $\mathcal{ALC}$-satisfiable. The implication in the other

direction is immediate since any crisp interpretation is a model of the additional TBox components $\mathcal{T}^*$:

Proposition 1. *C is satisfiable wrt. T in $\mathcal{ALC}$ iff $\pi(\mathrm{T}, C)$ is satisfiable in $\mathbb{ALC}$.*

It is straightforward to see that the many-one reduction $\pi(\mathrm{T}, C)$ can be computed in linear time (wrt. the size of T and C) for each finite set T of GCIs and concept expressions $C \in \mathcal{C}(\Sigma)$. As an immediate consequence we get the following corollary:

Corollary 1 (EXPTIME-Hardness of KB Satisfiability). *The problem of deciding the satisfiability of KBs in $\mathbb{ALC}$ is* EXPTIME*-hard if GCIs are allowed.*

EXPTIME*-Membership.* KB satisfiability in $\mathbb{ALC}$ is in EXPTIME since [18] shows that checking KB satisfiability in $\mathbb{ALC}$ can be reduced (in polynomial time) to checking KB satisfiability in $\mathcal{ALC}$ which is known to be in EXPTIME, since KB satisfiability is in EXPTIME even for an extension of $\mathcal{ALC}$, i.e. the more expressive DL $\mathcal{SHIQ}$ [21, Corollary 6.30].

Theorem 1 (EXPTIME-Completeness of KB Satisfiability). *The problem of deciding the satisfiability of KBs in $\mathbb{ALC}$ is* EXPTIME*-complete if GCIs are allowed.*

Besides KB satisfiability, one can show that also the following reasoning problems are EXPTIME-complete: $\mathcal{K} \models C \equiv D$, $\mathcal{K} \models C \sqsubseteq D$, concept disjointness wrt. $\mathcal{K}$, concept satisfiability wrt. $\mathcal{K}$, $\mathcal{K} \models \langle o : C \geq n \rangle$, $\mathcal{K} \models \langle o : C \leq n \rangle$, and $\mathcal{K} \models \langle o : C = n \rangle$; we refer the reader for more details to [6].

4 A Decision Procedure Based on Fuzzy Type Elimination

We present a decision procedure for KB satisfiability in $\mathbb{ALC}$ which does not rely on systematic search in the first place (as e.g. tableau-based methods), but instead constructs a canonical interpretation by means of a fixpoint construction. The so-constructed (canonical) interpretation (if non-empty) satisfies the TBox of a KB and allows to derive a model for the given knowledge base $\mathcal{K}$ iff $\mathcal{K}$ is satisfiable. In contrast to tableau-based procedures a canonical interpretation is in general *not* tree-shaped. Further, it can be shown that the number of iterations required to reach a fixpoint is *linear* in the modal depth of $\mathcal{K}$.

Preprocessing. Without loss of generality, we can restrict ourselves to *normalized* knowledge bases [16], i.e. knowledge bases which contain only fuzzy ABox assertions of the form $\langle \alpha \geq d \rangle$, by applying the following equivalent transformation fuzzy ABox axioms: $\langle i : C \leq d \rangle \rightsquigarrow \langle i : \neg C \geq 1 - d \rangle$ and $\langle i : C = d \rangle \rightsquigarrow \langle i : C \geq d \rangle$, $\langle i : \neg C \geq 1 - d \rangle$. Further, we can assume that all axioms in $\mathcal{K}$ are in box normal form (BNF) [10] (i.e. the only negative concept subexpressions are of the form $\neg \forall R.C$ or negated atomic concept names $\neg C$), by exhaustively applying the following equivalent transformation to concept expressions: $\neg(C \sqcap D) \rightsquigarrow \neg C \sqcup \neg D$, $\neg(C \sqcup D) \rightsquigarrow \neg C \sqcap \neg D$, and $\neg\neg C \rightsquigarrow C$. These preprocessing steps can be performed altogether in linear time wrt. the size of the input KB.

4.1 Basic Notions and Intuition

Types. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ denote a normalized $\mathbb{ALC}$ knowledge base in BNF. The *closure* of a knowledge base $\mathsf{cl}(\mathcal{K})$ is defined as the smallest set of concept expressions such that for all $C \in \mathsf{sub}(\mathcal{K})$, if C is not of the form $\neg D$, then $\{C, \neg C\} \subseteq \mathsf{cl}(\mathcal{K})$. Further, let $\mathsf{PossDeg}(\mathcal{K})$ denote the set of all relevant possibility degrees that can be derived from $\mathcal{K}$, i.e. $\mathsf{PossDeg}(\mathcal{K}) = \{0, 0.5, 1\} \cup \{d | \langle \alpha \geq d \rangle \in \mathcal{A}\} \cup \{1 - d | \langle \alpha \geq d \rangle \in \mathcal{A}\}$. It has been shown in [17,18] that if $\mathcal{K}$ is satisfiable, then there is as well a model of $\mathcal{K}$ which assigns possibility degrees in $\mathsf{PossDeg}(\mathcal{K})$ only. Hence, for our purposes we do not need to consider arbitrary possibility degrees $d \in [0, 1]$, but only the *finite* set $\mathsf{PossDeg}(\mathcal{K})$ that can be derived from $\mathcal{K}$.

We can then introduce the notion of a *type*, which allows to represent individuals of an interpretation in a syntactic way:

Definition 2 (Fuzzy $\mathcal{K}$-Type). *A* fuzzy $\mathcal{K}$-type τ *is a maximal subset of* $\mathsf{cl}(\mathcal{K}) \times \mathsf{PossDeg}(\mathcal{K})$ *such that the following conditions are satisfied: 1. if* $\langle C, d \rangle \in \tau$ *and* $\langle C, d' \rangle \in \tau$ *then* $d = d'$ *2. if* $C = \neg C'$ *then* $\langle C, d \rangle \in \tau$ *iff* $\langle C', 1 - d \rangle \in \tau$ *3. if* $C = C' \sqcap C''$ *then* $\langle C, d \rangle \in \tau$ *iff* $\langle C', d' \rangle \in \tau$ *and* $\langle C'', d'' \rangle \in \tau$ *and* $d = min(d', d'')$ *4. if* $C = C' \sqcup C''$ *then* $\langle C, d \rangle \in \tau$ *iff* $\langle C', d' \rangle \in \tau$ *and* $\langle C'', d'' \rangle \in \tau$ *and* $d = max(d', d'')$ *5. for all* $C \sqsubseteq C' \in \mathcal{T}$*: if* $\langle C, d \rangle \in \tau$ *and* $\langle C', d' \rangle \in \tau$ *then* $d \leq d'$ *6. if* $C = \top$ *then* $\langle C, 1 \rangle \in \tau$.

Since $\mathsf{cl}(\mathcal{K})$ and $\mathsf{PossDeg}(\mathcal{K})$ are both finite sets, there are at most $2^{|\mathsf{cl}(D)| \cdot |\mathsf{PossDeg}(\mathcal{K})|}$ different $\mathcal{K}$-types. Each type τ can be seen as an individual and syntactically represents *all* (fuzzy) properties that can be observed about that individual: $\langle C, d \rangle \in \tau$ indicates that the individual τ belongs to C with the possibility degree d. Hence, the set of all $\mathcal{K}$-types (or simply types) provides enough vocabulary to let us describe all kinds of interpretations for $\mathcal{K}$ simply by fixing how to interconnect individuals (and therefore types).

Canonical Model. It turns out that it is possible to connect types in a fixed (or canonical) way, such that the interconnection defined is consistent with *almost* all properties specified syntactically in the type. The interconnections can be derived from the types themselves:

For a set of types T we can define for each role R a *canonical accessibility relation* $\Delta_R : T \times T \rightarrow \mathsf{PossDeg}(\mathcal{K})$ that "maximally" interconnects types $\tau, \tau' \in T$ with possibility degree $d \in \mathsf{PossDeg}(\mathcal{K})$: let $\delta(d, d') := 1$ if $d \leq d'$ and $\delta(d, d') := 1 - d$ if $d > d'$. Then, we can define Δ_R by

$$\Delta_R(\tau, \tau') := min\{\delta(d, d') | \langle \forall R.C, d \rangle \in \tau, \langle C, d' \rangle \in \tau'\}$$

if $\forall R.C \in \mathsf{cl}(\mathcal{K})$ for some $C \in \mathbf{C}$, and $\Delta_R(\tau, \tau') := 1$ otherwise.

This way, we can construct a canonical interpretation $\mathcal{I}_T$ for any given set of types T using the canonical interconnection of types by Δ_R as follows: $\mathcal{I}_T = (T, \cdot^{\mathcal{I}_T})$ with (i) for any concept name C in $\mathcal{K}$ and any $\tau \in T$ we set $C^{\mathcal{I}_T}(\tau) = d$ if $\langle C, d \rangle \in \tau$, and (ii) $R^{\mathcal{I}_T}(\tau, \tau') = \Delta_R(\tau, \tau')$ for any role R in $\mathcal{K}$ and any $\tau, \tau' \in T$. Please note, that by our definition of $\mathcal{K}$-types, $\mathcal{I}_T$ is well-defined for any concept name or role name. However,

our definition deliberately leaves open the interpretation of individuals. We therefore define in fact a class of canonical interpretations, each of which fixes a specific way of how to interpret the individuals in a KB $\mathcal{K}$.

The canonical interconnection in $\mathcal{I}_T$ is chosen in such a way that all assignments of possibility degrees to concepts of the form $C = \forall R.C \in \tau$ are lower bounds for the possibility degrees that are in fact assigned by a canonical interpretation $\mathcal{I}_T$. Hence, such a canonical interpretation is *almost* immediately a (canonical) model for the terminology $\mathcal{T}$, i.e. it satisfies that

$$C^{\mathcal{I}_T}(\tau) = d \text{ iff } \langle C, d\rangle \in \tau \qquad (*)$$

for *almost* all $C \in \mathsf{cl}(\mathcal{K})$ and therefore $\mathcal{I}_T \models C \sqsubseteq C'$ for all $C \sqsubseteq C' \in \mathcal{T}$ by clause (5) in our definition of $\mathcal{K}$-types. That $(*)$ is satisfied by $\mathcal{I}_T$ is straightforward for the cases of concept names C, or complex concepts of the form $C = C' \sqcap C''$, $C = C' \sqcup C''$, $C = \neg C'$ and the $C^{\mathcal{I}_T}(\tau) \geq d$ case for $C = \forall R.C$ by our definition of types and the definition of Δ_R. The only cases where $(*)$ can be violated by $\mathcal{I}_T$ is for types τ containing universally role restricted concepts $\forall R.C$ that are assigned a possibility degree which is *too small* (wrt. the R-successor types τ' in $\mathcal{I}_T$) to properly reflect the semantics of $\forall R.C$ in $\mathbb{ALC}$, i.e. to coincide with the *greatest* lower bound of the set

$$\{max(1 - R^{\mathcal{I}_T}(\tau, \tau'), C^{\mathcal{I}_T}(\tau')) \mid \tau' \in T\}$$

Types τ in which the possibility degree assigned d to $\forall R.C$ is too small to be consistent with the semantics of $\mathbb{ALC}$ are called *bad types*. Bad types $\tau \in T$ can be detected easily, since they satisfy that there exist $R \in \mathbf{R}, C \in \mathcal{C}(\Sigma), d \in \mathsf{PossDeg}(\mathcal{K})$ s.t. $\langle \forall R.C, d\rangle \in \tau$ and for all $\tau' \in T$: if $\langle C, d'\rangle \in \tau'$ then $max(1 - \Delta_R(\tau, \tau'), d') > d$.

This suggests the following simple algorithm (which uses a *fuzzy type elimination* process at its core): in order to compute a maximal interpretation that satisfies all terminological axioms, we start off with the maximal set of types (i.e all $\mathcal{K}$-types) and iteratively fix all problems that prevent $(*)$ from being satisfied by removing bad types. This way, we must eventually reach a fixpoint after finitely many steps. If the resulting set of types is non-empty, we know that $(*)$ must hold (since all problems have been fixed) and therefore we can be certain that the corresponding canonical interpretation satisfies $\mathcal{T}$ (and covers all other possible models of $\mathcal{T}$ at the same time). Hence, we eventually need to check if all ABox axioms are satisfied by the canonical interpretation. If this is the case, we have found a model for $\mathcal{K}$, otherwise, we know that there can not be any interpretation that satisfies both $\mathcal{T}$ and $\mathcal{A}$ at the same time. In other words, $\mathcal{K}$ is not satisfiable.

4.2 Algorithm

The type elimination process sketched above can be formalized as shown in Algorithm 1. Note that the emptiness test for the fixpoint T is covered implicitly: if the fixpoint T is empty, then the test in the if-statement fails trivially.

The termination, soundness, and completeness of our algorithm can be proven formally (cf. [6] for full proofs):

Algorithm 1. The Type Elimination-based Decision Procedure $\mathbf{FixIt}(\mathbb{ALC})$

procedure satisfiable($\mathcal{K}$): boolean
T := $\{\tau | \tau$ is a $\mathcal{K}$-type $\}$;
repeat
  $T' := T$;
  $T := T' \setminus badtypes(T')$;
until $T = T'$;
if *there exists a total function* $\pi : Ind_{\mathcal{A}} \rightarrow T$ *s.t.* $\langle C, d' \rangle \in \pi(o)$ *and* $d \leq d'$ *for each* $\langle o : C \geq d \rangle \in \mathcal{A}$, *and* $\Delta_R(\pi(o), \pi(o')) \geq d$ *for each* $\langle R(o, o') \geq d \rangle \in \mathcal{A}$ **then**
  return true;
end
return false;

function $badtypes(T) : 2^T$
return $\{\tau \in T | \langle \forall R.C, d \rangle \in \tau$ and for all $\tau' \in T$: if $\langle C, d' \rangle \in \tau'$ then $max(1 - \Delta_R(\tau, \tau'), d') > d\}$;

Theorem 2 (Termination). *For any* $\mathbb{ALC}$ *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *the algorithm* $\mathbf{FixIt}(\mathbb{ALC})$ *terminates after finitely many steps with either* true *or* false *as return value.*

The following lemma is a key element of the soundness and completeness proof and shows that by successively removing bad types we can indeed ensure that types encode possibility degree assignments to concepts that coincide with the canonical interpretation, and that any such canonical interpretation is a model of the $\mathcal{T}$.

Let T be the set of types that is computed as the fixpoint in the algorithm $\mathbf{FixIt}(\mathbb{ALC})$, i.e. $badtypes(T) = \emptyset$ and let $\mathcal{I}_T = (T, \cdot^{\mathcal{I}_T})$ be a canonical interpretation for T as defined above.

Lemma 1. *For each* $\mathcal{K}$*-type* τ, *concept* $C \in \mathsf{cl}(\mathcal{K})$ *and* $d \in \mathsf{PossDeg}(\mathcal{K})$ *it holds that* $C^{\mathcal{I}_T}(\tau) = d$ *iff* $\langle C, d \rangle \in \tau$. *Further,* $\mathcal{I}_T \models \mathcal{T}$.

Theorem 3 (Soundness). *If* $\mathbf{FixIt}(\mathbb{ALC})$ *returns true for a* $\mathbb{ALC}$ *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, *then* $\mathcal{K}$ *is satisfiable.*

A second key element for the completeness proof is the following lemma that shows that our canonical way of interconnecting types (in the fixpoint set) is maximal or the strongest possible one in the following sense: the interconnection R of individuals o, o' defined by any model $\mathcal{I}$ of $\mathcal{K}$ is covered by the canonical interconnection Δ_R of the respective types $\tau(o), \tau(o')$ representing o, o' in $\mathcal{I}$.

Lemma 2. *Let* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *be any model of* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. *For each individual* $o \in \Delta^{\mathcal{I}}$ *we define its corresponding type* $\tau(o) := \{\langle C, d \rangle \in \mathsf{cl}(\mathcal{K}) \times \mathsf{PossDeg}(\mathcal{K}) | C^{\mathcal{I}}(o) = d\}$. *Then,* $\Delta_R(\tau(o), \tau(o')) \geq R^{\mathcal{I}}(o, o')$ *for all* $o, o' \in \Delta^{\mathcal{I}}$.

Theorem 4 (Completeness). *If an* $\mathbb{ALC}$ *knowledge base* $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *is satisfiable, then* $\mathbf{FixIt}(\mathbb{ALC})$ *returns true for* $\mathcal{K}$.

This leads to the main result, which is an immediate consequence of Theorems 3, 4, and 2:

Corollary 2. *The algorithm* $\mathbf{FixIt}(\mathbb{ALC})$ *is a sound and complete decision procedure for knowledge base satisfiability in* $\mathbb{ALC}$.

4.3 Runtime and Space Requirements

We now analyze the runtime and space requirements of our algorithm based on a naive implementation model to derive upper bounds. The considered implementation works directly on types and explicit representations of sets of types.

In the following we assume that the number p of different possibility degrees that can occur in any KB $\mathcal{K}$ is fixed. This assumption seems realistic and does not restrict applications of FDLs in practice, i.e., we can assume a limited (numerical) resolution of sensors and algorithms (or humans) assigning possibility degrees to individual observations. Further, we consider the computation of the basic functions min, max, $1 - d$ and comparisons $d \leq d'$ as atomic operations with unit costs.

Representation. According to Def. 2, a $\mathcal{K}$-type τ is a function $\tau : \mathsf{cl}(\mathcal{K}) \to \mathsf{PossDeg}(\mathcal{K})$ that satisfies a particular consistency property. Fix some total order $\langle C_1, C_2, \ldots, C_k \rangle$ on the subset $\mathsf{cl}_+(\mathcal{K})$ of $\mathsf{cl}(\mathcal{K})$ that consists of all positive concept expression $C_i \in \mathsf{cl}(\mathcal{K})$. Then, we can represent a type τ by a sequence of pairs

$$\tau = \langle d_1, \overline{d_1} \rangle, \langle d_2, \overline{d_2} \rangle, \ldots, \langle d_k, \overline{d_k} \rangle$$

with $k = |\mathsf{cl}_+(\mathcal{K})| \in O(|\mathcal{K}|)$, where d_i represents the possibility degree assigned to a positive concept subexpression $C_i \in \mathsf{cl}(\mathcal{K})$ and $\overline{d_i}$ represents the possibility degree assigned to a negative concept subexpression $\neg C_i \in \mathsf{cl}(\mathcal{K})$. There are only finitely many relevant possibility degrees. Assuming a binary encoding $\langle \cdot \rangle_{01} : \mathsf{PossDeg}(\mathcal{K}) \to \{0,1\}^p$ of possibility degrees, each such sequence requires at most $2 \cdot k \cdot log_2(p) \in O(k)$ bits. Clearly, not each such sequence represents a $\mathcal{K}$-type. However, for any sequence checking the consistency requirements from Def. 2 can be done time $O(k \cdot |\mathcal{T}|)$ and space $O(1)$: property (1) is satisfied already by our encoding, properties (2)-(3) and (6) can each be decided in $O(k)$, property (5) requires $O(k \cdot |\mathcal{T}|)$ computation steps. In any case, we need only $O(1)$ additional space for the computation.

The algorithm can be separated into three different phases: the initialization step, the fixpoint computation, and the final test. We analyze all of these steps one-by-one.

Initialization Phase. To compute the set of all $\mathcal{K}$-types, we generate any sequence $\langle d_1, \overline{d_1} \rangle, \langle d_2, \overline{d_2} \rangle, \ldots, \langle d_k, \overline{d_k} \rangle$ s.t. $d_i, \overline{d_k} \in \mathsf{PossDeg}(\mathcal{K})$. For each sequence this requires at most $2k$ steps. Testing if such a generated sequence satisfies the consistency requirements, takes at most $O(k \cdot |\mathcal{T}|)$ steps and only constant additional space. Any sequence that satisfies the consistency requirements is stored in a linked list. During the initialization phase, we need to check p^{2k} sequence, which requires $O(p^{2k} \cdot k \cdot |\mathcal{T}|)$ computation steps and $O(p^{2k} \cdot k)$ space.

Fixpoint Computation. The current set of types T in the fixpoint computation is represented as linked list from which elements are removed during this phase. Clearly, the function $\delta(d, d')$ can be computed in constant time. The computation of $\Delta_R(\tau, \tau')$ therefore can be performed in $O(k^2)$ computation steps and constant additional space for any two given types τ, τ'. Given a type $\tau \in T$, we can then determine if τ is a bad type (wrt. T) in $O(k \cdot (|T| \cdot k^2)) = O(k^3 \cdot |T|)$ basic computation steps using constant space only. This test has to be performed for all $|T|$ types in T in order to compute T'. Removing a bad type from the current list of types T requires $O(|T|)$ time and constant additional space. To find out if we reached a fixpoint after an iteration, we can simply use a boolean variable, which is set to $false$ at the beginning of each loop and set to $true$ if a type is removed from the list. This requires at most $O(|T|)$ additional assignments and one boolean comparison during each iteration and constant additional space. Hence, each iteration of the loop (to compute T') requires $O((k^3 + 1) \cdot |T|^2 + |T|)$ computation steps and constant additional space.

Let t denote the size of the initial set of types T. Since we need at most t iterations to reach a fixpoint, we can compute the fixpoint in at most $O((k^3 + 1) \cdot t^3 + t)$ steps using $O(t)$ additional memory units. Since $t \leq p^{2k}$, we can derive $O((k^3 + 1) \cdot p^{6k} + p^{2k})$ as an upper bound on the number of computation steps performed for the fixpoint computation and $O(p^{2k})$ as an upper bound for the additionally required space.

Final Test. We can represent a total total mapping $\pi : Ind_{\mathcal{A}} \rightarrow T$ as a mapping from $Ind_{\mathcal{A}}$ to the index of an element in the list T. This requires, $O(i \cdot log_2(t))$ additional memory units with $i = |Ind_{\mathcal{A}}|$ and lookups for values of π can be performed in $O(t)$ time using a hashing function and a subsequent iteration over the list representing T. Given such a mapping π, we can therefore determine if the required property in the if-statement in Algorithm 1 is satisfied by π in at most $O(a \cdot (t \cdot k^2))$ steps (where $a = |\mathcal{A}|$) using $O(1)$ additional memory only. We can generate any such mapping $\pi : Ind_{\mathcal{A}} \rightarrow T$ in at most $O(i \cdot log_2(t))$ steps. Further, there are at most t^i different such mapping, which we can generate and test one-by-one. Hence, we can implement that final test using at most $O(t^i \cdot (i \cdot log_2(t) + a \cdot (t \cdot k^2))) = O(p^{2ki} \cdot (i \cdot k \cdot log_2(p) + a \cdot (p^{2k} \cdot k^2)))$ steps and $O(i \cdot k \cdot log_2(p))$ additional memory units.

Overall runtime and space bounds. For an upper bound on the overall execution of the algorithm, we can sum up the runtime and space bounds for the three different phases above: In regard of runtime, the algorithm requires no more than $O(p^{2k} \cdot k \cdot |\mathcal{T}| + (k^3 + 1) \cdot p^{6k} + p^{2k} + (p^{2ki} \cdot (i \cdot k \cdot log_2(p) + a \cdot (p^{2k} \cdot k^2))))$ computation steps. In regard of the space, the algorithm requires no more than $O(p^{2k} \cdot k + p^{2k} + i \cdot k \cdot log_2(p) + |\mathcal{K}|)$ space. Since $k, i \in O(|\mathcal{K}|)$ and since we assume that p is a constant (and hence independent from a size of a KB $\mathcal{K}$), the algorithm requires at most exponentially many computations steps and an exponential amount of memory in regard of the size of the input KB $\mathcal{K}$.

This means that (under the realistic assumptions) our algorithm can be implemented in a way that is *worst-case optimal* in regard of the runtime of the algorithm, since by Theorem 1 the problem of determining the satisfiability of a KB in $\mathbb{ALC}$ requires exponential time (wrt. the size of the input KB) in the worst case. Further, note that applying a standard tableau-based decision procedure (e.g. [1]) to (crisp) equisatisfiable $\mathcal{ALC}$-reduction of a general $\mathbb{ALC}$ KB usually yields only a NEXPTIME-upper-bound

on the runtime of such an (indirect) decision procedure. Therefore, indirect reasoning approaches by reduction to classical tableau-based methods usually can only give *substantially worse* runtime guarantees.

Consequently, the major obstacles when using the algorithm in practice are (i) the exponential space requirement and (ii) the necessity to consider all types in $\tau \in T$ during each loop one-by-one. These problems are mainly based on the fact that we use *explicit* representations of set of types. A potential solution to these problems is known from the area of Symbolic Model Checking [8] and has already been successfully applied for the implementation of the $\mathcal{KBDD}$ procedure in [10]: the use *implicit* (or declarative) representations of sets of types by means of formulae and to implement set-theoretic operations by formula modifications and (un)satisfiability tests. In particular, set-theoretic operations on sets can be implemented as set-at-a-time operations instead of element-at-a-time operations which can speed up computation significantly.

5 Discussion: Reasoning with Terminologies in Fuzzy Tableau-Based Methods

Tableau-methods can be used to determine the satisfiability of KBs. In order to detect the satisfiability status of a given KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, they construct tree-structured labeled graphs (i.e. completion graphs). The nodes in a completion graph represent individuals in an interpretation $\mathcal{I}$ for $\mathcal{K}$ and the edges capture the interrelation of individuals via roles in $\mathcal{K}$. Node labels capture (bounds on) the degree to which the individual represented by the node is a member of a concept, edge labels specify (bounds on) the degree of interrelation of the connected individuals wrt. roles. The completion graph is initialized to represent the ABox $\mathcal{A}$. Then, it is stepwise extended by analyzing concept labels of nodes and interrelations. An expansion can create new labels for nodes and edges and insert new nodes and edges into the graph. The extension process follows a set of so-called completion rules which are exhaustively applied. During the completion process non-deterministic choices can appear: there might be more than one way to extend the graph further, but we can not tell which of these extensions will eventually lead to a successful construction of a model for $\mathcal{K}$ (if there is one). In order to stay complete, all possible choices (for any node and any node label) need to be considered as long as no model has been constructed yet. The latter can be determined by checking labels for elementary contradictions (in regard of the specified bounds on membership degrees). If no completion rules are applicable anymore and there are no labels to nodes and edges which contain an elementary contradiction (i.e. we have a fully-expanded and clash-free completion graph), then we found in fact a (witness for a) model for $\mathcal{K}$. If no such graph can be constructed at all, $\mathcal{K}$ is unsatisfiable.

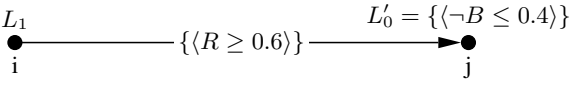
Example. Consider the ABox $\mathcal{A} = \{\langle i : \forall R.\neg B \sqcup \exists R.(B \sqcap C) \leq 0.4\rangle, \langle i : \forall R.C \geq 0.7\rangle\}$. The completion process starts with the completion graph

$$L_0 = \{\langle \forall R.\neg B \sqcup \exists R.(B \sqcap C) \leq 0.4\rangle, \langle \forall R.C \geq 0.7\rangle\}$$

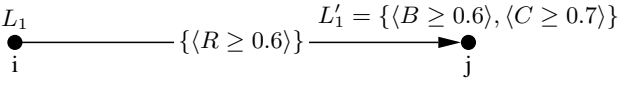
Since i can be a member of $\forall R.\neg B \sqcup \exists R.(B \sqcap C)$ only up to degree 0.4, we can derive upper bounds on the possibility degree for i wrt. $\forall R.\neg B$ and $\exists R.(B \sqcap C)$ directly from the semantics of the concept constructor $\sqcup$ and update the label by two new constraints

$$L_1 = L_0 \cup \{\langle \forall R.\neg B \leq 0.4\rangle, \langle \exists R.(B \sqcap C) \leq 0.4\rangle\}$$
$$\bullet$$
$$\text{i}$$

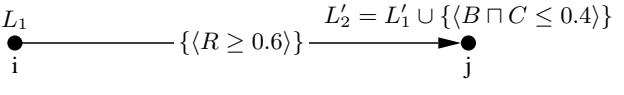
To satisfy $\langle \forall R.\neg B \leq 0.4\rangle$, we introduce a new R-successor node j, a constraint on the degree of R-interrelation between i and j, and a constraint on the degree of membership of j in $\neg B$. Both constraints follow immediately from the semantics of the universal role restriction in $\mathbb{ALC}$:



To satisfy the constraint $\langle \forall R.C \geq 0.7\rangle \in L_1$ we add $\langle C \geq 0.7\rangle$ to L'_0. Further, in $\mathbb{ALC}$ $\langle \neg B \leq 0.4\rangle \in L'_0$ can be rewritten as $\langle B \geq 0.6\rangle$



Finally, the only constraint that we did not consider yet is $\langle \exists R.(B \sqcap C) \leq 0.4\rangle \in L_0$. To ensure the satisfaction of this concept expression, we need to add $\langle B \sqcap C \leq 0.4\rangle$ (since the edge constraint $\langle R \geq 0.6\rangle$ and the upper bound $\langle R \leq 0.4\rangle$ are inconsistent with each other):



Considering the semantics of the concept intersection in $\mathbb{ALC}$, $\langle B \sqcap C \leq 0.4\rangle$ is satisfied iff $\langle B \leq 0.4\rangle$ *or* $\langle B \leq 0.4\rangle$ are satisfied. Hence, we face a non-deterministic choice point in the completion process. Since all possible extensions lead to an unsatisfiable constraint systems node j in the completion graph, we can conclude that the given ABox $\mathcal{A}$ is unsatisfiable.

Tableau-based implementations deal with (or) non-deterministic choice points by backtracking. Since for any choice, an unsatisfiable node might be detected only after a lot of node label and graph extension steps, backtracking can become a very costly operation.

Integration of GCIs. In order to integrate GCIs into the completion graph extension process sketched above, standard tableau-based methods (e.g. [16], and similarly [7]) exploit the following observation: An interpretation $\mathcal{I}$ satisfies the GCI $C \sqsubseteq D$ (i.e. $\mathcal{I} \models C \sqsubseteq D$) iff for all (relevant) possibility degrees $n \in \mathsf{PossDeg}(\mathcal{K})$ and all individuals $i \in \Delta^{\mathcal{I}}$ either $\langle i : C < n\rangle$ or $\langle i : D \geq n\rangle$ holds.

To reflect this property in the tableau completion process, a non-deterministic rule for the extension of node labels is added to the inference system: for each node i in the

completion graph, each GCI $C \sqsubseteq D$ in the TBox, and each (relevant) possibility degree n, we either insert the constraint $\langle i : C < n \rangle$ or $\langle i : D \geq n \rangle$ into the current node label.

It is obvious that using this new inference rule GCIs essentially become the *major source of non-determinism* for tableau-based inference procedures: for *each* node i appearing (*somewhen*) in the completion graph, we have a distinct alternative for *each* relevant possibility degree $n \in \mathsf{PossDeg}(\mathcal{K})$ and *each* GCI $C \sqsubseteq D \in \mathcal{T}$. Hence, for each node i on a path of length l in the completion graph, we get up to $|\mathsf{PossDeg}(\mathcal{K})| \cdot |\mathcal{T}|$ different alternatives. This makes up to $l^{|\mathsf{PossDeg}(\mathcal{K})| \cdot |\mathcal{T}|}$ distinct cases that need to be considered during the completion process in the worst-case. It is further known [1] that when supporting GCIs, the maximal path length in the completion graph can be limited only by an upper bound that is (itself) exponential in the size of the input ABox $\mathcal{A}$ (using a technique called *blocking*). This gives a doubly-exponential runtime for tableau-based methods in the worst case, such that without clever implementation techniques and good heuristics for guiding the backtracking search, a tableau-based reasoner integration GCIs as described above might not be usable even in rather small practical scenarios.

In this paper, we present a method that does not suffer from this problem, i.e. no non-deterministic choice-points are introduced.

6 Related Work

Our method $\mathbf{FixIt}(\mathbb{ALC})$ generalizes the principle (i.e. a type elimination process) underlying the top-down variant of the $\mathcal{KBDD}$ procedure proposed in [10] for the modal logic $\mathbf{K}$ to the (more expressive) FDL $\mathbb{ALC}$. Further, our method integrates (fuzzy) ABoxes and TBoxes in the inference process both of which are *not* dealt with in $\mathcal{KBDD}$.

Inference Algorithms for FDLs and Reasoning with GCIs. So far, reasoning in Fuzzy DLs has been mostly based on tableau-methods (e.g., [7,15,16,17]). Most of these methods do not support reasoning with general terminologies as it is possible with $\mathbf{FixIt}(\mathbb{ALC})$. The first method ever to integrate GCIs into FDL reasoning is [16]. A very similar approach is presented in [7] for the fuzzy variant of a more expressive DL, namely $\mathcal{SHI}$. Very recently, [20] proposed a novel and elegant method for reasoning with GCIs (under a more general semantics than here) which is inspired by earlier works on tableau-based reasoning in multi-valued logics. The method combines a tableau-construction procedure with a Mixed Integer Linear Programming (MILP) solver that serves as an oracle to the FDL tableau procedure. To the best of our knowledge there is no other approach to deal with GCIs in FDLs available at present. $\mathbf{FixIt}(\mathbb{ALC})$ therefore represents an interesting enrichment of inference calculi toolbox for FDLs, since no non-determinism is introduced by considering GCIs. A similar effect is achieved in [20] by the substantial modification of a standard tableau-based method and an extension with an MILP oracle: the tableau-expansion process does not become non-deterministic by introducing GCIs. However, depending on the solution techniques applied inside the MILP solver, non-determinism might simply be shifted from the tableau-construction process into the MILP oracle. In such cases, respective computational inefficiencies

would then simply be hidden in then MILP oracle, but not actually resolved. A very similar approach that is not fixed to a specific semantics is presented in [3].

Further, [18] demonstrates how to use inference procedures for *classical* DLs to perform reasoning in (some) FDLs. This allows to use algorithms that have been developed for classical DLs in FDL reasoning (for some FDLs) in an indirect way. Please note that the $\mathcal{KBDD}$ procedure can not be used in such an indirect way to perform $\mathbb{ALC}$ reasoning, since both TBoxes and ABoxes are not supported.

[4,5] consider a fuzzy version of $\mathcal{ALC}$ using arbitrary continuous t-norms (and the corresponding residuated implications) to define the semantics of the concept constructors and proposes a method for deciding $\emptyset \models \langle C \sqsubseteq D \geq 1\rangle$ and the satisfiability of $\langle C \sqsubseteq D \geq 1\rangle$ by mapping to a (decidable) propositional fuzzy logic. The generated propositional problems can be exponentially bigger than the FDL input problem. Although, the semantics considered in [4,5] is more general than here (but differs for universal role restrictions), the proposed decision procedures cover more limited reasoning tasks, i.e. no background knowledge $\mathcal{K}$ is considered.

7 Conclusions and Future Work

We presented a novel procedure $\mathbf{FixIt}(\mathbb{ALC})$ for deciding knowledge base (KB) satisfiability in the FDL $\mathbb{ALC}$, introducing a *new class* of inference procedures into FDL reasoning. Besides the tableau-based methods [3,7,16,20], it is the only (and the first non-tableau-based) approach to integrate general terminologies in FDL reasoning that we are aware of.

Additionally, we clarified the worst-case complexity of the reasoning problem that is solved by the algorithm and showed that deciding the satisfiability of a KB in $\mathbb{ALC}$ is an EXPTIME-complete problem. A discussion of a (straigthforward) implementation of the algorithm based on explicit representations of types shows that our algorithm can be implemented in a way that is worst-case optimal wrt. its runtime.

The main research questions that we want to address next are as follows: we will study means of implicit representation of sets of fuzzy types known from Symbolic Model Checking [8], in particular their implementation by means of Ordered Binary Decision Diagrams (OBDDs) [2] similar to [10], therefore addressing the main obstacle to apply the procedure in practice. A major question concerning optimization is clearly how to implement the final test of the algorithm efficiently, e.g. by heuristic search using the information in the ABox effectively to find the required mapping. The integration of optimizations such as full vs. lean representations or particle vs. types as discussed in [10] should be straightforward. We want to evaluate the effectiveness of the method by an implementation and comparison to tableau-based systems for FDLs. Moreover, we believe that it is interesting to study a bottom-up variant of $\mathcal{KBDD}$ in the context of FDLs too, and to check if the integration of ABoxes can be done more efficiently in such a variant. Finally, we would like to see to what extend the method can cover other semantics for FDLs (e.g. other t-norms) and extended constructs, such as fuzzy modifiers and concrete domains.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Bryant, R.E.: Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Comput. Surv. 24(3), 293–318 (1992)
3. Haarslev, V., Pai, H., Shiri, N.: Uncertainty Reasoning for Ontologies with General TBoxes in Description Logic. In: Costa, P.C.G., D'Amato, C., Fanizzi, N., Laskey, K.B., Laskey, K., Lukasiewicz, T., Nickles, M., Pool, M. (eds.) Uncertainty Reasoning for the Semantic Web I. LNCS (LNAI). Springer, Heidelberg (2008)
4. Hájek, P.: Making fuzzy description logic more general. Fuzzy Sets and Systems 154(1), 1–15 (2005)
5. Hájek, P.: What does Mathematical Fuzzy Logic offer to Description Logic? In: Fuzzy Logic and the Semantic Web. Capturing Intelligence. Elsevier, Amsterdam (2006)
6. Keller, U., Heymans, S.: On Fixpoint-based Decision Procedures for Fuzzy Description Logics I. Technical Report STI TR 2008-08-07, Semantic Technology Institute (STI), University of Innsbruck (August 2008), `http://www.uwekeller.net/publications.html`
7. Li, Y., Xu, B., Lu, J., Kang, D.: Discrete Tableau Algorithms for $\mathcal{FSHI}$. In: Proceedings of the International Workshop on Description Logics, DL (2006)
8. McMillan, K.L.: Symbolic Model Checking. Kluwer Academic Publishers, Norwell (1993)
9. Meghini, C., Sebastiani, F., Straccia, U.: A model of multimedia information retrieval. Journal of the ACM 48(5), 909–970 (2001)
10. Pan, G., Sattler, U., Vardi, M.Y.: BDD-based decision procedures for the modal logic K. Journal of Applied Non-Classical Logics 16(1-2), 169–208 (2006)
11. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL Web Ontology Language Semantics and Abstract Syntax. Candidate Recommendation August 18, 2003, W3C (2003)
12. Pratt, V.R.: A Near-Optimal Method for Reasoning about Action. J. Comput. Syst. Sci. 20(2), 231–254 (1980)
13. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI 1991), pp. 466–471 (1991)
14. Schmidt-Schauß, M., Smolka, G.: Attributive Concept Descriptions with Complements. Artif. Intell. 48(1), 1–26 (1991)
15. Stoilos, G., Stamou, G.B., Pan, J.Z., Tzouvaras, V., Horrocks, I.: Reasoning with very expressive fuzzy description logics. J. Artif. Intell. Res (JAIR) 30, 273–320 (2007)
16. Stoilos, G., Straccia, U., Stamou, G., Pan, J.: General Concept Inclusions in Fuzzy Description Logics. In: Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI 2006), pp. 457–461. IOS Press, Amsterdam (2006)
17. Straccia, U.: Reasoning within Fuzzy Description Logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
18. Straccia, U.: Transforming Fuzzy Description Logics into Classical Description Logics. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS, vol. 3229, pp. 385–399. Springer, Heidelberg (2004)
19. Straccia, U.: A Fuzzy Description Logic for the Semantic Web. In: Sanchez, E. (ed.) Fuzzy Logic and the Semantic Web, Capturing Intelligence, ch. 4, pp. 73–90. Elsevier, Amsterdam (2006)
20. Straccia, U., Bobillo, F.: Mixed integer programming, general concept inclusions and fuzzy description logics. Mathware & Soft Computing 14(3), 247–259 (2007)
21. Tobies, S.: Complexity results and practical algorithms for logics in Knowledge Representation. Phd thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)

Quantitative Comparison of Intuitionistic and Classical Logics - Full Propositional System*

Antoine Genitrini[1] and Jakub Kozik[2]

[1] PRiSM, CNRS UMR 8144, Université de Versailles Saint-Quentin
45 av. des États-Unis, 78035 Versailles cedex, France
antoine.genitrini@prism.uvsq.fr
[2] Theoretical Computer Science, Jagiellonian University,
ul. Łojasiewicza 6, 30-348 Kraków, Poland
jakub.kozik@uj.edu.pl

Abstract. We address the problem of quantitative comparison of classical and intuitionistic logics within the language of the full propositional system. We apply two different approaches, to estimate the asymptotic fraction of intuitionistic tautologies among classical tautologies, obtaining the same results for both. Our results justify informal statements such as "about 5/8 of classical tautologies are intuitionistic".

1 Introduction

It is a standard approach to use the notion of density [6,1] to analyse quantitative relations between countable sets. The general idea is to consider subsets of elements of bounded size, and to observe the uniform measure of one subset in the other when the maximal allowed size tends to infinity. This approach requires that the number of elements of bounded size is finite.

One of the first papers to address the quantitative aspects of intuitionistic logic was [6], which (according to the authors) was partially motivated by the short note in some paper of Statman saying:"*It is a good bet but not a sure thing, that ρ (type) contains a closed term*". Most results of that paper were formulated in terms of inhabitation of types in simple λ-calculus. However, under Curry-Howard isomorphism (see e.g. [8]), they translate directly to the framework of intuitionistic logic.

The authors of [6] considered calculus with a finite number of ground types, and only functional types. In terms of logical formulae it means that the number of different variables in a formula was bounded by some constant, and the only allowed connective was $\Rightarrow$. The authors proved that at least 1/3 of classical

* Research described in this paper was partially supported by the A.N.R. project *SADA*, French government research grant for young scientists (program number 0185) and by POLONIUM grant *Quantitative research in logic and functional languages*, cooperation between Jagiellonian University of Krakow, L' École Normale Supérieure de Lyon and L'Université de Versailles Saint-Quentin, contract number 7087/R07/R08.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 280–294, 2009.

tautologies are intuitionistic and gave some lower and upper bounds (dependent on the number of allowed variables) for the density of intuitionistic tautologies among all the formulae. They have also conjectured that, among the formulae with the number of different variables bounded by any constant, the probability that a classical tautology of size n, chosen uniformly at random, is intuitionistic, tends to one, when n goes to infinity. The conjecture was known to be true for the formulae using only one variable for the trivial reason that both sets of tautologies are equal in this case.

Although the conjecture of [6] is false, a slight reformulation turned out to be true. The authors of [3] proved that the lower bound for the density of intuitionistic logic in the classical one tends to 1, when the number of allowed variables tends to infinity. This counter-intuitive result raised a question about the appropriateness of the approach. In fact, the assumption about the bounded number of variables seemed to have a strong influence on the result. In the paper [4] the authors suggested another approach, in which formulae was considered up to a renaming of variables (i.e. two formulae which differ only in the naming of variables were assumed to be equal). In that case the authors could deal with formulae with an unbounded number of variables, while preserving the property that there is only a finite number of formulae of bounded size. In that setup, using methods similar as in [3], the authors obtained an analogous result - the density is equal to 1. We want to emphasize at this point that the fact that both results coincide is in our opinion no less surprising that the fact that the densities tend to 1.

The work presented in this paper is a continuation of this research, considering other languages of propositional formulae. Among them the most interesting is the language which admits all the usual connectives $\Rightarrow, \wedge, \vee$, and the constant $\bot$. We prove that in this case the coherence of the results in both approaches is preserved, even though the limit is no longer equal to 1, but to 5/8.

2 Prerequisites and Results

For any set of finite elements A and $n \in \mathbb{N}$ we denote by $A(n)$ the number of elements of set A with size n (the element is finite if it has finite size).

Formulae and Terms. Let $\mathrm{Var} = \{x_1, x_2, x_3, \ldots\}$ be a countable set of variables, $\bot$ be a constant, and $\mathcal{C} = \{\Rightarrow, \vee, \wedge\}$ be a set of binary connectives. A term in our system is a binary complete tree with internal nodes labelled by the elements of $\mathcal{C}$ and leaves labelled by the elements of $\mathrm{Var} \cup \{\bot\}$ (precisely the tree is rooted and planar i.e. the order of descendants matters). For every $k \in \mathbb{N}$ let $\mathcal{F}_k$ denote the set of terms whose variables belong to the set $\mathrm{Var}_k = \{x_1, \ldots, x_k\}$. The set of all terms is denoted by Term. The size of a tree is its number of leaves.

Two terms are *α-equivalent* if they differ only in the naming of variables, i.e. $(\varphi, \psi) \in \alpha$ if there exists injective relabelling function $r : \mathrm{Var} \to \mathrm{Var}$, such that we obtain ψ after relabelling variables from φ according to r. Clearly, α is an equivalence relation. We denote Term/α by $\mathcal{F}_\infty$. We use the name formula both for terms and for elements from $\mathcal{F}_\infty$.

Intuitionistic Logic. For a general reference about intuitionistic logic we suggest e.g. [8]. These are well known facts that every intuitionistic tautology is classical, and that the converse is not true (even for implicational fragment). In our proofs we use also the fact that the Heyting algebra of open subsets of $\mathbb{R}$ (with respect to euclidean topology) is a complete model for the propositional intuitionistic logic.

2.1 Main results

Let $Cl, Int \subset \text{Term}$ denote the sets of terms which are respectively classical and intuitionistic tautologies. For every $k \in \mathbb{N}$ we put

$$Cl_k = Cl \cap \text{Term}_k, \qquad Int_k = Int \cap \text{Term}_k,$$

and

$$Cl_\infty = Cl/\alpha, \qquad Int_\infty = Int/\alpha.$$

Let a sequence $(d_k(n))_{n\in\mathbb{N}}$ be defined as: $d_k(n) = Int_k(n)/Cl_k(n)$. Each fraction $d_k(n)$ equals the probability that a formula, chosen uniformly at random among the set of elements of Cl_k of size n, is an intuitionistic tautology. If the sequence converges, its limit is denoted by D_k and is called the (relative) density of Int_k in Cl_k. We do not address the problem of existence of D_k. We use the following bounds instead:

$$D_k^- = \liminf_{n\to\infty} d_k(n), \qquad \text{and} \qquad D_k^+ = \limsup_{n\to\infty} d_k(n).$$

The first of our main results says:

$$\lim_{k\to\infty} D_k^- = \lim_{k\to\infty} D_k^+ = \frac{5}{8}.$$

This is analogous to the approach taken in [3] for the implicational fragment. In that case the limit was 1.

Considering the formulae "up to the names of variables" enables an arbitrary number of different variables in formula, while preserving the property that there is only a finite number of formulae with bounded size. In this approach we consider the sequence $(d_\infty(n))_{n\in\mathbb{N}}$ defined as follows: $d_\infty(n) = Int_\infty(n)/Cl_\infty(n)$.

The second of our main results says that

$$\lim_{n\to\infty} d_\infty(n) = \frac{5}{8}.$$

We could give an informal interpretation that "about $\frac{5}{8}$ of classical tautologies are intuitionistic". It was proved in [4] that the analogous approach for the implicational fragments gives the density 1.

We derive both results in an unified way from some structural properties of tautologies.

2.2 Structure and Labelling

For every formula φ, *the structure of* φ is a binary tree constructed from φ by forgetting about the labelling of its leaves (e.g. by changing it so that each leaf is labelled by $\bullet$). The definition can be naturally extended to the formulae from $\mathcal{F}_\infty$, since all the terms in each equivalence class have the same structure. The set of structures in our system is denoted by $\mathcal{T}$. It is the set of binary complete trees with internal nodes labelled by $\Rightarrow, \wedge$ or $\vee$ and all leaves labelled by $\bullet$.

We say that a node is an $\Rightarrow$-node if the node is labelled with $\Rightarrow$. We use an analogous convention for the other connectives.

For a formula $\varphi \in \mathcal{F}_k$ with n leaves, a leaf labelling of φ is a function $f : \{1, \ldots, n\} \to \mathrm{Var}_k \cup \{\bot\}$ such that $f(i)$ coincides with the label at the i-th leaf of φ. We call such a function a k-labelling of size n.

For a formula $[\varphi] \in \mathcal{F}_\infty$ with n leaves, a leaf labelling of $[\varphi]$ is the equivalence relation R on the set $\{0, 1, \ldots, n\}$ consisting of all the pairs of numbers of leaves which are labelled by the same symbol (variable or $\bot$) and all the pairs $(0, j), (j, 0)$ for each leaf j labelled with $\bot$. Note that the relation R does not depend on the chosen representative of the equivalence class $[\varphi]$. It contains information about which leaves are labelled by the same variable (but not by which variable), and which leaves are labelled with $\bot$. We call such a relation a ∞-labelling of size n.

As usual the size of a formula is the number of its leaves. We use the same convention for the size of a structure. We denote by $\mathcal{T}(n)$ the number of trees from $\mathcal{T}$ of size n.

Note, that in all the considered cases (bounded for every $k \in \mathbb{N}$ and unbounded) we have a one-to-one correspondence between the structure-labelling pairs of the size n and the formulae of that size. That fact is reflected in simple expressions for the numbers of formulae of size n. We have

$$\mathcal{F}_k(n) = \mathcal{T}(n)(k+1)^n, \qquad \mathcal{F}_\infty(n) = \mathcal{T}(n)B(n+1), \tag{1}$$

where $B(n+1)$ is the number of equivalence relations on the set $\{0, 1, \ldots, n\}$, known as Bell number (see e.g. [5]).

2.3 Generating Functions

Within this paper we make an extensive use of the theory of generating functions and analytic combinatorics (see [2]). All the generating functions in this paper are ordinary ones.

We use a notation which always exposes the formal parameters of a generating function. E.g. we write $g(z)$ instead of g for some generating function $\sum_{n\in\mathbb{N}} g_n z^n$. Although the notation may be a little bit misleading it provides a convenient way of expressing substitutions for formal parameters. It is a standard convention to denote by $[z^n]g(z)$ the coefficient g_n (for the function $g(z)$ defined as above).

One of the most basic generating functions in this paper is the one enumerating all the structures. We denote it by $t(z)$. By a standard constructions we get

an algebraic equation for $t(z)$, where z marks the size:

$$t(z) = 3t(z)^2 + z.$$

Solving this equation (and choosing the proper solution) we get

$$t(z) = (1 - \sqrt{1 - 12z})/6.$$

The radius of convergence of $t(z)$ is $\rho = \frac{1}{12}$, $t(z)$ is bounded within its circle of convergence, and $t(\rho) = \lim_{z \to_{\mathbb{R}} \rho^-} t(z) = \frac{1}{6}$.

Lemma 1. *Let $f, g \in \mathbb{Z}[[z]]$ be algebraic generating functions, having a common unique dominating singularity at $\varrho \in \mathbb{R}_+$. Suppose that these functions have Puiseux expansions around ϱ of the form*

$$f(z) = c_f + (z - \varrho)^{\frac{1}{2}} (d_f + o(1)), \quad g(z) = c_g + (z - \varrho)^{\frac{1}{2}} (d_g + o(1)).$$

with both d_f, d_g being nonzero. Then: $\lim_{n \to \infty} \frac{[z^n]f(z)}{[z^n]g(z)} = \lim_{z \to_{\mathbb{R}} \varrho^-} \frac{f'(z)}{g'(z)}$.

Using Theorem VII.8 from [2] we obtain this equality because both sides are equal to d_f/d_g.

3 Structural Properties of Tautologies

Within this section we consider structural properties of tautologies, which are independent of the approach we use (bounded or unbounded). In order to obtain results independent from the kind of labelling, we use $\mathcal{F}$ to denote the set of formulae under consideration, and the function $Lab : \mathbb{N} \to \mathbb{N}$ which for every $n \in \mathbb{N}$ returns the number of all different labellings of the structure of size n. In particular we get results for the unbounded approach by setting $\mathcal{F}$ equal to $\mathcal{F}_\infty$ and $Lab(n) = B(n+1)$. In an analogous way the results are translated to the bounded case for every fixed number of variables k by substituting $\mathcal{F}$ with $\mathcal{F}_k$ and $Lab(n)$ with $(k+1)^n$. E.g. in this convention equations (1) are formulated as

$$\mathcal{F}(n) = \mathcal{T}(n) Lab(n).$$

Pointed Structures. An m-pointed structure is a pair (t, s) of a structure t and a sequence of m different leaves of t. Usually we use a pointed structure to encode some constraints on the allowed labellings. For example let A denote some set of 1-pointed structures and consider the set of formulae $\mathcal{F}_A$, which can be constructed from elements of A by the labellings which assign $\perp$ to the pointed leaf. For every structure $a \in A$ of size n we are free to label all the remaining leaves. Therefore, there are $Lab(n-1)$ labellings which give a formula from $\mathcal{F}_A$ from the structure a. Therefore $\mathcal{F}_A(n) \leqslant A(n) Lab(n-1)$.

Tree Decomposition. We say that a node v in a tree $t \in \mathcal{T}$ is *k-shallow* if the path from the root to v goes at most k times to the left from a node labelled with $\Rightarrow$. We say it is a *k-layer* node if it is k-shallow but not $(k-1)$-shallow.

To obtain an upper bound for the number of tautologies we focus on 3-shallow leaves.

Let us consider the set of trees $P \subset \mathcal{T}$ such that every left subtree of every node labelled with $\Rightarrow$ is a leaf (i.e. all 1-layer nodes are leaves). Let $p(t,u)$ be the generating function for such trees with t marking leaves which are left sons of $\Rightarrow$-node, and u marking the remaining leaves (t denotes a formal parameter, not the generating function for all trees which we denote by $t(z)$). The generating function is given implicitly by an initial condition and by the equation

$$p(t,u) = t \cdot p(t,u) + 2p(t,u)^2 + u, \tag{2}$$

which reflects the fact that every such a tree is either an implication with the left subtree being a 1-layer leaf and the right subtree belonging to P, or a conjunction or a disjunction with both subtrees belonging to P, or a leaf (which is a 0-shallow leaf).

Clearly, $p(t(z), uz)$ is the generating function of all structures, with z marking the size and u marking 0-shallow leaves. We define a sequence of generating functions:

$$p_{\leqslant 0}(t,u) = t \qquad p_{\leqslant n+1}(t,u) = p(p_{\leqslant n}(t,u), u).$$

Each function $p_{\leqslant n}(t,u)$ is the generating function of the set of structures in which all $(n+1)$-layer nodes are leaves, with u marking n-shallow leaves, and t marking leaves which are left sons of n-layer $\Rightarrow$-nodes (i.e. all $(n+1)$-layer leaves). Since every node in every tree is an i-layer node for exactly one i, we get for every $n \in \mathbb{N}$, $t(z) = p_{\leqslant n}(t(z), z)$.

Proposition 1. *For $s, m \in \mathbb{N}$ let $\mathcal{T}^{(m)}_{\leqslant s}$ denote the set of m-pointed structures with all pointed leaves being s-shallow (we call them s-shallow m-pointed structures). There exists a positive constant $c_{s,m} \in \mathbb{R}$ such that*

$$\lim_{n\to\infty} \frac{\mathcal{T}^{(m)}_{\leqslant s}(n)}{\mathcal{T}(n)} = c_{s,m}.$$

Proof. Solving the equation (2) and using the fact that $p(0,0) = 0$ we get

$$p(t,u) = \frac{1}{4}(1 - t - \sqrt{(1-t)^2 - 8u}).$$

It shows that the function $p(t,u)$ is holomorphic in the set $D_\varepsilon = \{(t,u) \in \mathbb{C}^2 : |t| \leqslant \frac{1}{6} + \varepsilon,\ |u| \leqslant \frac{1}{12} + \varepsilon\}$ for some small positive $\varepsilon \in \mathbb{R}$ (note that $t(\rho) = \frac{1}{6}$ and $\rho = \frac{1}{12}$). By non-negativity of the coefficients of the expansion of $p(t,u)$ at 0 we get that $\max_{(t,u)\in D_0} |p(t,u)| = p(\frac{1}{6}, \frac{1}{12}) = \frac{1}{6}$. Therefore each $p_{\leqslant s}(t,u)$ is holomorphic in D_ϵ (for some positive $\epsilon \in \mathbb{R}$) and so are all its partial derivatives, in particular $\frac{\partial^m p_{\leqslant s}(t,u)}{(\partial u)^m}$. The latter function is exactly the generating function

of s-shallow m-pointed structures in which all $(m+1)$-layer nodes are leaves (marked with variable t). It remains to substitute the generating function of all structures for t to obtain the generating function for all s-shallow m-pointed structures. We substitute u with z so that the variable z marks all the leaves (after pointing we are no longer interested in s-shallow leaves). As a result we obtain a function

$$p_{m,s}(z) = \frac{\partial^m p_{\leqslant s}(t,u)}{(\partial u)^m}|_{u:=z,t:=t(z)},$$

which is the generating function of the set of all s-shallow m-pointed structures. Let $\widehat{D_\epsilon}$ denote the set $D_\epsilon \setminus [\rho, \infty]$. Then the function $t(z)$ is analytically continuable to the set $\widehat{D_\epsilon}$, and since the outer function is holomorphic in $\widehat{D_\epsilon}$ we know that the function $p_{m,s}(z)$ is analytically continuable to that set. On the other hand the combinatorial interpretation shows that $p_{m,s}(z)$ must have singularity in ρ. Therefore we know that $p_{m,s}(z)$ has unique dominating singularity in ρ. In fact we know also that the limit $\lim_{z\to\mathbb{R}\rho^-} p_{m,s}(z) < \infty$, therefore the singularity is not a pole. Since $p_{m,s}(z)$ is algebraic, the singularity must be a branching point. By the fact that $t(v^2)$ is analytic at ρ we get that $p_{m,s}(v^2)$ is analytic as well, which shows that the branching type of $p_{m,s}(z)$ at ρ is 2 (we excluded the existence of pole). Finally, the fact that $\lim_{z\to\mathbb{R}\rho^-} p'_{m,s}(z) = \infty$ shows that the singularity is of the square root type. A straightforward application of the Lemma 1 proves the result.

In fact we need the Proposition 1 only for the sets $\mathcal{T}^{(2)}_{\leqslant 3}, \mathcal{T}^{(3)}_{\leqslant 3}, \mathcal{T}^{(4)}_{\leqslant 3}$, and the results for these sets can be easily established by explicit calculations of their generating functions.

Shallow Repetitions. For every formula φ and set of its leaves L we say that φ has r repetitions among the leaves from L if r equals the difference between the cardinality of L and the number of different variables assigned to the leaves from L. If the set L is the set of k-shallow leaves we say that φ has r k-shallow repetitions. Note, that the occurrence of the constant is treated as a repetition e.g. the formula $x \Rightarrow \bot$ has one repetition among all its leaves.

Proposition 2. *Within the set of elements of $\mathcal{F}$ of size n, the fraction of formulae with at least two 3-shallow repetitions is asymptotically bounded from above by $c\frac{Lab(n-2)}{Lab(n)}$. Formally, let $\mathcal{F}^{[\geqslant 2]}_{\leqslant 3}$ denote the set of formulae with at least two 3-shallow repetitions, we have*

$$\frac{\mathcal{F}^{[\geqslant 2]}_{\leqslant 3}(n)}{\mathcal{F}(n)} \lesssim c\frac{Lab(n-2)}{Lab(n)}$$

Proof. Every formula $\varphi \in \mathcal{F}^{[\geqslant 2]}_{\leqslant 3}$ satisfies at least one of the following properties:

A . φ contains two 3-shallow leaves labelled with $\bot$;
B . φ contains one 3-shallow leaf labelled with $\bot$ and two 3-shallow leaves labelled by the same variable;

C . φ contains three 3-shallow leaves labelled by the same variable;
D . two variables occur at least twice among 3-shallow leaves of φ.

Let $\mathcal{F}^A, \mathcal{F}^B, \mathcal{F}^C, \mathcal{F}^D$ denote the sets of formulae from $\varphi \in \mathcal{F}^{[\geqslant 2]}_{\leqslant 3}$ with the previous properties. Clearly

$$\mathcal{F}^{[\geqslant 2]}_{\leqslant 3}(n) \leqslant \mathcal{F}^A(n) + \mathcal{F}^B(n) + \mathcal{F}^C(n) + \mathcal{F}^D(n),$$

and the inequality is usually strict.

Every formula from $\mathcal{F}^A$ contains at least two 3-shallow leaves labelled with $\bot$. Therefore it can be constructed from a 3-shallow 2-pointed structure by some labelling which assigns $\bot$ to the pointed leaves. Hence

$$\mathcal{F}^A(n) \leqslant \mathcal{T}^{(2)}_{\leqslant 3}(n) \cdot Lab(n-2).$$

An analogous reasoning for the other sets gives:

$$\mathcal{F}^B(n) + \mathcal{F}^C(n) \leqslant 2 \cdot \mathcal{T}^{(3)}_{\leqslant 3}(n) \cdot Lab(n-2),$$

$$\mathcal{F}^D(n) \leqslant \mathcal{T}^{(4)}_{\leqslant 3}(n) \cdot Lab(n-2).$$

Using these equations and Proposition 1 we obtain

$$\frac{\mathcal{F}^{[2]}_{\leqslant 3}(n)}{\mathcal{F}(n)} \leqslant \frac{(\mathcal{T}^{(2)}_{\leqslant 3}(n) + 2\mathcal{T}^{(3)}_{\leqslant 3}(n) + \mathcal{T}^{(4)}_{\leqslant 3}(n))}{\mathcal{T}(n)} \frac{Lab(n-2)}{Lab(n)}$$
$$\sim (c_{2,3} + 2c_{3,3} + c_{4,3}) \cdot \frac{Lab(n-2)}{Lab(n)}.$$

The above proposition will be used to show that we can neglect all formulae with at least two 3-shallow repetitions, since the number of them will be shown to be essentially smaller than the number of tautologies.

Simple Classical Tautologies. For a formula φ let a boolean valuation v^1_φ assign $True$ only to those variables which have occurrences on the first layer, and $v^{1,3}_\varphi$ only to those which have occurrences on the first or third layer. The following proposition is a consequence of the fact that if there is no 1-shallow repetitions in φ, then the formula is valuated to $False$ by v^1_φ.

Proposition 3. *If a formula φ does not contain at least one 1-shallow repetition, it is not a classical tautology.*

Definition 1. *A* positive path *in a formula (tree) is a path from the root to some node, which never crosses a $\wedge$-node, and never goes left from a $\Rightarrow$-node. A node is called positive if there exists a positive path to it (see Fig. 1).*

Definition 2. *A* negative path *in a formula (tree) is a path from the root, which contains a positive $\Rightarrow$-node h, such that the path is going left from h and then follows only $\wedge$-nodes (if any). A node is called negative if there exists a negative path to it (see Fig. 1).*

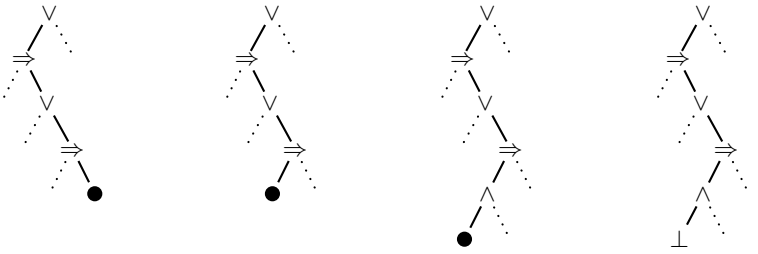


Fig. 1. From left to right: a positive path, not a positive path, a negative path, a tree with a negative leaf labelled with $\bot$

For every formula it is enough to valuate one of its positive nodes to $True$ or one of its negative nodes to $False$, to ensure that the valuation of the whole formula is $True$.

These two definitions give rise to two large families of classical tautologies.

Observation 1. *All the formulae in which some negative leaf is labelled with* $\bot$ *are* classical tautologies. *The set of those formulae is denoted by* $S_\bot$ *(see Fig. 1).*

Observation 2. *All the formulae in which some positive leaf is labelled by the same variable as some negative leaf are classical tautologies. We denote this family by* S_R.

We call the formulae from the set $S_R \cup S_\bot$, *simple tautologies*. We focus on the formulae with exactly one 2-shallow repetition and no more than one 3-shallow repetition. The set of such formulae is denoted by $\mathcal{H}$. In the next two propositions we show that all tautologies belonging to $\mathcal{H}$ are simple.

Proposition 4. *If a formula* $\varphi \in \mathcal{H} \setminus S_\bot$ *contains a 3-shallow leaf* l *labelled with* $\bot$*, then it is not a tautology.*

Proof. If the leaf l is not 1-shallow then there are no 1-shallow repetitions and the boolean function defined by the formula is not a tautology (Proposition 3). If l is 0-shallow then we can use the valuation v^1_φ which valuates all the 0-shallow leaves to $False$ and all the 1-layer leaves to $True$. In that case the formula is valuated by v^1_φ to $False$.

In the remaining case l is a 1-layer leaf but is not negative. Let s be the last $\vee$-node or $\Rightarrow$-node on the path from the root to l. The node s is a 1-layer node, because l is not negative.

Suppose that s is labelled by $\vee$. One of its subtrees does not contain shallow occurrences of l. In that subtree all the 0-shallow leaves are valuated by $v^{1,3}_\varphi$ to $True$ (because they are all 1-layer in φ) therefore the whole subtree with root s is valuated to $True$ by $v^{1,3}_\varphi$.

If s is labelled by $\Rightarrow$ then let s_2 be its left son. Clearly s_2 is a 2-layer node. Since we have only one 3-shallow repetition and it is realized by a 1-shallow node labelled with $\bot$, all the labels of 2-layer and 3-layer leaves are not repeated among the 3-shallow leaves. Therefore the valuation $v^{1,3}_\varphi$ assigns $False$ to all the

2-layer leaves, and $True$ to all the 3-layer leaves. Consequently, every 2-layer node is valuated to $False$. Is means that also s_2 is valuated to $False$, but then s is valuated to $True$.

In both cases the only 1-layer nodes which are valuated by $v_\varphi^{1,3}$ to $False$ are below the node s, which is valuated to $True$ anyway. Hence every 1-layer node which is a left son of a 0-shallow node is valuated to $True$. But then all 0-shallow nodes are valuated to $False$, which proves that φ is not a classical tautology.

Proposition 5. *If a formula $\varphi \in \mathcal{H} \setminus S_R$ contains a variable repetition, then it is not a tautology.*

Proof. If φ does not contain any 1-shallow repetition, then according to the Proposition 3, the formula does not define a constant function. If both leaves with repeated variable are on the same level, then the valuation v_φ^1 valuates all the 0-shallow leaves to $False$ and all 1-layer leaves to $True$, and the formula is valuated by v_φ^1 to $False$.

Let l_1, l_2 be the 3-shallow leaves labelled with the same variable. We can assume that l_1 is 0-shallow and l_2 is a 1-layer leaf. If l_1 is not positive then there exists a node s on the path from the root to l_1, which is labelled with $\wedge$. In that case the only 0-shallow nodes which can be valuated to $True$ by v_φ^1 are below s. But s is valuated to $False$, because it is a $\wedge$-node and one of its subtrees is valuated by v_φ^1 to $False$ (the one which does not contain l_1).

In the remaining case we have two leaves l_1, l_2 labelled with the same variable, such that l_1 is positive (and hence 0-shallow), l_2 is not negative but is a 1-layer leaf. In this case we use boolean valuation b which assigns $False$ only to those variables which have occurrences among 0-shallow or 2-layer leaves. Then the leaf l_2 is valuated to $False$ and we can use the same reasoning as in the case when some not negative 1-layer leaves is labelled with $\bot$, to prove that φ is not a tautology.

We have

$$S_\bot(n) + S_R(n) - \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n) \leqslant Cl(n) \leqslant S_\bot(n) + S_R(n) + \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n) \qquad (3)$$

The lower bound comes from the fact that every formula which belongs to $S_\bot \cap S_R$ has at least two 3-shallow repetitions. The upper bound is a consequence of Propositions 4 and 5, which together say that all tautologies which are not simple belong to $\mathcal{F}_{\leqslant 3}^{[\geqslant 2]}$.

Simple Intuitionistic Tautologies. It is easy to show that all the formulae from $S_\bot$ are intuitionistic tautologies. This is not true for S_R, and a simple counterexample is $x \vee (x \Rightarrow y)$. However, we can prove the following proposition (e.g. by using the Heyting algebra of open subsets of $\mathbb{R}$).

Proposition 6. *A formula from $S_R \cap \mathcal{H}$ is an intuitionistic tautology if and only if the positive prefix of the path leading to the negative leaf with the repeated variable is a prefix of the path leading to the positive leaf with the repeated variable (i.e. the last common node is a $\Rightarrow$-node). The set of those formulae is denoted by S_{RI} (see Fig. 2).*

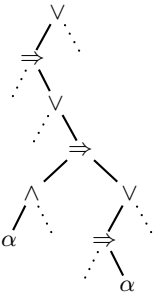


Fig. 2. A tree with a negative path and a positive path with the same prefix

Analogously to the inequality (3) we get

$$S_{\perp}(n) + S_{RI}(n) - \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n) \leqslant Int(n) \leqslant S_{\perp}(n) + S_{RI}(n) + \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n). \quad (4)$$

4 Counting Simple Families

Within this section we denote by $\mathcal{T}_{\leq 3}^{(2,3,4)}(n)$ the value $\mathcal{T}_{\leqslant 3}^{(2)}(n) + 2\mathcal{T}_{\leqslant 3}^{(3)}(n) + \mathcal{T}_{\leqslant 3}^{(4)}(n)$. For any $i \in \mathbb{N}$, a i-positive-pointed structure is a i-pointed structure, whose pointed leaves are all positive (note that positivity of leaves depends only on the structure). Negative-pointed structures are defined analogously. We use the following sets of structures:

- $\mathcal{T}_N$ - the set of 1-negative-pointed structures,
- $\mathcal{T}_{PN}$- the set of 2-pointed structures such that the first pointed leaf is positive and the second one is negative,
- $\mathcal{T}_{\widehat{PN}}$- the subset of $\mathcal{T}_{PN}$ consisting of all the structures for which the positive prefix of the path to the negative pointed leaf is a prefix of the (positive) path to the positive pointed leaf.

In the following propositions we give bounds on the number of elements of $S_{\perp}$ and S_R of size n.

Proposition 7.

$$\mathcal{T}_N(n) \cdot Lab(n-1) - \mathcal{T}_{\leq 3}^{(2,3,4)}(n) \cdot Lab(n-2) \leqslant S_{\perp}(n) \leqslant \mathcal{T}_N(n) \cdot Lab(n-1).$$

Proof. From every 1-negative-pointed structure we can construct a formula from $S_{\perp}$ by a labelling which assigns $\perp$ to the pointed leaf. If the pointed structure has n leaves we have exactly $Lab(n-1)$ such labellings. Since every formula from $S_{\perp}$ can be constructed in this way we get:

$$S_{\perp}(n) \leqslant \mathcal{T}_N(n) Lab(n-1).$$

The inequality is usually strict, since some formulae can be generated with more than one structure-labelling pair of considered type. Those are exactly the formulae, that have at least two negative leaves labelled with $\perp$ (hence they have

at least two 3-shallow repetitions). But the number of pairs which generate formulae with that property is smaller than the number of pairs which generate all the formulae with at least two 3-shallow repetition. We get (just as in the proof of Proposition 2),

$$\mathcal{T}_N(n)Lab(n-1) - \mathcal{T}_{\leq 3}^{(2,3,4)}(n) \cdot Lab(n-2) \leqslant S_\perp(n).$$

An analogous result holds for S_R.

Proposition 8.

$$\mathcal{T}_{PN}(n) \cdot Lab(n-1) - \mathcal{T}_{\leq 3}^{(2,3,4)}(n) \cdot Lab(n-2) \leqslant S_R(n),$$

$$S_R(n) \leqslant \mathcal{T}_{PN}(n) \cdot Lab(n-1)$$

We omit the technical proof.

Corollary 1. *Applying the same reasoning for S_{RI} as in Proposition 8, we get both following inequalities*

$$\mathcal{T}_{\widehat{PN}}(n) \cdot Lab(n-1) - \mathcal{T}_{\leq 3}^{(2,3,4)}(n) \cdot Lab(n-2) \leqslant S_{RI}(n),$$

$$S_{RI}(n) \leqslant \mathcal{T}_{\widehat{PN}}(n) \cdot Lab(n-1).$$

4.1 Structural Limits

To prove our main results we need to calculate the following three "structural limits":

$$D_N = \lim_{n\to\infty} \frac{\mathcal{T}_N(n)}{\mathcal{T}(n)}, \quad D_{PN} = \lim_{n\to\infty} \frac{\mathcal{T}_{PN}(n)}{\mathcal{T}(n)}, \quad D_{\widehat{PN}} = \lim_{n\to\infty} \frac{\mathcal{T}_{\widehat{PN}}(n)}{\mathcal{T}(n)}.$$

Proposition 9. $D_N = \lim_{n\to\infty} \frac{\mathcal{T}_N(n)}{\mathcal{T}(n)} = \frac{5}{8}$.

Proof. Let $T(z)$ be the generating function for $\mathcal{T}$ and $g_N(y,z)$ be the generating function for $\mathcal{T}$, with z marking the size and y marking the leaves that can be obtained from root by paths containing only $\wedge$-nodes. It satisfies:

$$g_N(y,z) = 2T(z)^2 + g_N(y,z)^2 + yz.$$

Let $f_N(y,z)$ be the generating function for all structures with z marking the size and with negative leaves marked with y. We have

$$f_N(y,z) = f_N(y,z)^2 + g_N(y,z)f_N(y,z) + T(z)^2 + z. \tag{5}$$

The first term is obtained when the root is labelled by $\vee$. The second one, by $\Rightarrow$, and the third term corresponds to $\wedge$.

By a classical construction (pointing corresponds to differentiation), to obtain the generating function for 1-negative-pointed structures $SN(z)$ it is enough to differentiate $f_N(y,z)$ with respect to the variable y, and then to substitute y by 1

(we no longer need bivariate function). Algebraic calculations and the application of the Lemma 1 give:

$$\lim_{n\to\infty}\frac{\mathcal{T}_N(n)}{\mathcal{T}(n)}=\lim_{n\to\infty}\frac{[z^n]SN(z)}{[z^n]T(z)}=\lim_{z\to_{\mathbb{R}}\frac{1}{12}^-}\frac{SN'(z)}{T'(z)}=\frac{5}{8}.$$

In the similar way we prove the following two propositions.

Proposition 10. $D_{PN}=\lim_{n\to\infty}\frac{\mathcal{T}_{PN}(n)}{\mathcal{T}(n)}=\frac{11}{8}$.

Proposition 11. $D_{\widehat{PN}}=\lim_{n\to\infty}\frac{\mathcal{T}_{\widehat{PN}}(n)}{\mathcal{T}(n)}=\frac{5}{8}$.

Using the bounds from the Proposition 7 and the limits we have computed, we get:

$$\frac{S_\perp(n)}{\mathcal{F}(n)}\leqslant\frac{\mathcal{T}_N(n)}{\mathcal{T}(n)}\cdot\frac{Lab(n-1)}{Lab(n)}\sim\frac{5}{8}\frac{Lab(n-1)}{Lab(n)}$$

and

$$\begin{aligned}\frac{S_\perp(n)}{\mathcal{F}(n)}&\geqslant\frac{\mathcal{T}_N(n)}{\mathcal{T}(n)}\frac{Lab(n-1)}{Lab(n)}-\frac{\mathcal{T}^{(2,3,4)}_{\leq 3}(n)}{\mathcal{T}(n)}\frac{Lab(n-2)}{Lab(n)}\\&\sim\frac{5}{8}\frac{Lab(n-1)}{Lab(n)}-C\frac{Lab(n-2)}{Lab(n)},\end{aligned}$$

for some $C\in\mathbb{R}$. Analogous estimates (using values D_{PN} and $D_{\widehat{PN}}$) hold for S_R and S_{RI}.

4.2 Main Results

Unbounded Case. The asymptotic behaviour of the Bell numbers is known due to the result of Moser and Wyman [7]. We are going to use the following property: $B(n-2)/B(n)=o(B(n-1)/B(n))$. The estimates from the previous subsection specialize to the unbounded case; inequalities 3 and 4 gives:

$$\begin{aligned}\frac{Int_\infty(n)}{\mathcal{F}_\infty(n)}&=\frac{S_\perp(n)+S_{RI}(n)}{\mathcal{F}_\infty(n)}+o\left(\frac{B(n)}{B(n+1)}\right)\\&\sim\frac{B(n)}{B(n+1)}\left(\frac{10}{8}+o(1)\right),\\\frac{Cl_\infty(n)}{\mathcal{F}_\infty(n)}&=\frac{S_\perp(n)+S_R(n)}{\mathcal{F}_\infty(n)}+o\left(\frac{B(n)}{B(n+1)}\right)\\&\sim\frac{B(n)}{B(n+1)}\left(\frac{16}{8}+o(1)\right)\end{aligned}$$

therefore

$$\frac{Int_\infty(n)}{Cl_\infty(n)}\sim\frac{5}{8}.$$

Bounded Case. We specialize now to the case with the number of variables bounded by k. We get

$$\limsup_{n\to\infty} \frac{S_\perp(n)}{\mathcal{F}(n)} \leqslant \frac{5}{8k}$$

and

$$\liminf_{n\to\infty} \frac{S_\perp(n)}{\mathcal{F}(n)} \geqslant \frac{5}{8k} - \frac{C}{k^2}.$$

Analogous reasoning for families S_R and S_{RI} gives

$$\limsup_{n\to\infty} \frac{S_R(n)}{\mathcal{F}(n)} \leqslant \frac{11}{8k} \qquad \liminf_{n\to\infty} \frac{S_R(n)}{\mathcal{F}(n)} \geqslant \frac{11}{8k} - \frac{C}{k^2},$$

and

$$\limsup_{n\to\infty} \frac{S_{RI}(n)}{\mathcal{F}(n)} \leqslant \frac{5}{8k} \qquad \liminf_{n\to\infty} \frac{S_{RI}(n)}{\mathcal{F}(n)} \geqslant \frac{5}{8k} - \frac{C}{k^2}.$$

Therefore we get

$$\begin{aligned}\limsup_{n\to\infty} \frac{Int_k(n)}{Cl_k(n)} &\leqslant \frac{\limsup_{n\to\infty} \mathcal{F}_k(n)^{-1}(S_\perp(n) + S_{RI}(n) + \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n))}{\liminf_{n\to\infty} \mathcal{F}_k(n)^{-1}(S_\perp(n) + S_R(n) - \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n))} \\ &= \frac{\frac{10}{8k} - o(\frac{1}{k})}{\frac{2}{k} + o(\frac{1}{k})} \sim_k \frac{5}{8}\end{aligned}$$

and

$$\begin{aligned}\liminf_{n\to\infty} \frac{Int_k(n)}{Cl_k(n)} &\geqslant \frac{\liminf_{n\to\infty} \mathcal{F}_k(n)^{-1}(S_\perp(n) + S_{RI}(n) - \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n))}{\limsup_{n\to\infty} \mathcal{F}_k(n)^{-1}(S_\perp(n) + S_R(n) + \mathcal{F}_{\leqslant 3}^{[\geqslant 2]}(n))} \\ &= \frac{\frac{10}{8k} - o(\frac{1}{k})}{\frac{2}{k} + o(\frac{1}{k})} \sim_k \frac{5}{8}.\end{aligned}$$

Hence

$$\lim_{k\to\infty} D_k^- = \lim_{k\to\infty} D_k^+ = \frac{5}{8},$$

which is the second of our main results.

5 Final Remarks

The reasoning we used for the full propositional system is also appropriate for other sets of connectives. If we allow only implication the method we presented reconstructs the results from [3] and [4] (i.e. in this case the density of intuitionistic logic in classical is 1). Adding conjunction and $\perp$ to the system does not change the situation. However, it suffices to consider the language which uses only $\Rightarrow$ and $\vee$ to observe a difference. For this language the asymptotic density of intuitionistic logic in the classical one equals 3/13.

Finally, we want once again to emphasize that the coherence of the results in the bounded and unbounded approaches is quite an interesting fact in itself. We believe that Proposition 1 sheds some light on this phenomenon.

References

1. Chauvin, B., Flajolet, P., Gardy, D., Gittenberger, B.: And/or trees revisited. Combinatorics, Probability & Computing 13 (2004)
2. Flajolet, P., Sedgewick, R.: Analytic combinatorics, in preparation, preprint (2008), http://algo.inria.fr/flajolet/Publications/book.pdf
3. Fournier, H., Gardy, D., Genitrini, A., Zaionc, M.: Classical and intuitionistic logic are asymptotically identical. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 177–193. Springer, Heidelberg (2007)
4. Genitrini, A., Kozik, J., Zaionc, M.: Intuitionistic vs classical tautologies, quantitative comparison. In: Miculan, M., Scagnetto, I., Honsell, F. (eds.) TYPES 2007. LNCS, vol. 4941, pp. 100–109. Springer, Heidelberg (2008)
5. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete mathematics: a foundation for computer science. Addison-Wesley Longman Publishing Co., Boston (1989)
6. Moczurad, M., Tyszkiewicz, J., Zaionc, M.: Statistical properties of simple types. Mathematical Structures in Computer Science 10(5), 575–594 (2000)
7. Moser, L., Wyman, M.: An asymptotic formula for the bell numbers. Transactions of the Royal Society of Canada XLIX (1955)
8. Sorensen, M., Urzyczyn, P.: Lectures on the curry-howard isomorphism (1998)

Tableaux and Hypersequents for Justification Logic

Hidenori Kurokawa

Department of Philosophy
The City University of New York, Graduate Center
365 Fifth Avenue, New York, NY 10016
hkurokawa@gc.cuny.edu

Abstract. Justification Logic is a new generation of epistemic logics which along with the traditional modal knowledge/belief operators also consider justification assertions 't is a justification for F.' In this paper, we introduce a prefixed tableau system for one of the major logics of this kind S4LPN, which combines the logic of proofs (LP) and epistemic logic S4 with an explicit negative introspection principle $\neg t:F \rightarrow \Box\neg t:F$. We show that the prefixed tableau system for S4LPN is sound and complete with respect to Fitting-style semantics. We also introduce a hypersequent calculus HS4LPN and show that HS4LPN is complete as far as we confine ourselves to a case where only a single formula is to be proven. We establish this fact by using a translation from the prefixed tableau system to the hypersequent calculus. This completeness result gives us a semantic proof of cut-admissibility for S4LPN under the aforementioned restriction.

1 Introduction

The first system of Justification Logic, the Logic of Proofs (LP), is introduced by Artemov ([1]) as a logic that can explicitly talk about proofs. An earlier sketch of the Logic of Proofs was suggested by Gödel in [7]. Several variants have been studied in combination with traditional modal logics. One such variant is S4LP, which was introduced by Artemov and Nogina ([2]) and also studied by Fitting ([5]). This logic contains knowledge modality $\Box F$ and justification assertions $t:F$. Other examples are LPP ([10]) and GLA ([8]), both of which are combination of LP and provability logic (GL, or Gödel-Löb logic). Artemov and Nogina in [2][1] introduced both logics, S4LP and S4LPN, using Hilbert-style axiomatic systems. The latter is S4LP with explicit negative introspection $\neg t:F \rightarrow \Box\neg t:F$. Fitting[5] and Renne[9] found destructive tableau systems for S4LP. But so far no tableau system or sequent calculus for S4LPN or GLA has been proposed. Moreover, since such a formula as $\Box t : F \vee \Box\neg t : F$ is a theorem of S4LPN and GLA, the task of finding cut-free destructive tableau systems for these logics seems to be hopeless. In this paper we suggest more flexible frameworks to

[1] We mostly follow the notation and the terminology of [2] concerning S4LPN.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 295–308, 2009.

give complete proof-systems for S4LPN. We first give a prefixed tableau system for S4LPN in the sense of [4]. However, from a philosophical perspective, one might think that a prefixed tableau system contain too much semantic information in the form of prefixes (cf., Avron [3]). To overcome this potential weakness, we formulate a hypersequent calculus for the logic (HS4LPN) and show that there is a way of converting a closed prefixed tableau to a proof in HS4LPN. This gives us a restricted but sufficiently strong version of the completeness theorem for HS4LPN without Cut that allows provability of all valid *formulas*. As a corollary, we will obtain a semantic proof of cut-admissibility for HS4LPN.

2 Prefixed Tableau System for S4LPN and Its Semantics

We present the language of S4LPN and a Hilbert-style axiom system for S4LPN. Proof terms and formulas in the language are as follows.[2]

$t ::= x|a|!t|t_1 \cdot t_2|t_1 + t_2$
$A ::= P_i|\bot|A_1 \to A_2|A_1 \wedge A_2|A_1 \vee A_2|t{:}A|\Box A$

The Hilbert-style axiomatic system of S4LPN given in [2] is as follows.

Axioms 0) Axioms of Propositional Logic
1) Axioms of explicit knowledge:
1. $t{:}F \to F$
2. $t{:}(F \to G) \to (s{:}F \to t{\cdot}s{:}G)$
3. $t{:}F \to !t{:}t{:}F$
4. $t{:}F \to t + s{:}F$, $s{:}F \to t + s{:}F$

2) Axioms of implicit knowledge:
1. $\Box F \to F$
2. $\Box(F \to G) \to (\Box F \to \Box G)$
3. $\Box F \to \Box\Box F$

3) Connecting Axiom: 1. $t{:}F \to \Box F$ 2. $\neg t{:}F \to \Box\neg t{:}F$

Rules of Inference 1. Modus Ponens 2. Necessitation $F/\Box F$
3. $c{:}A$ (Axiom Necessitation), where A is one of the above axioms

Now we present the prefixed tableau system. We use Fitting's terminology for basic notions in [4] and [6]. In particular, our prefix is a finite sequence of positive integers that has only 1 as the initial element of a sequence. In $\sigma T\varphi$ or $\sigma F\varphi$, σ is a prefix of a signed formula $T\varphi$ or $F\varphi$. σ' is accessible from σ iff $\sigma \leq \sigma'$ ($\leq$ means that "is an (not necessarily proper) initial segment of"). σ' is e-accessible from σ iff $1 \leq \sigma$ and $1 \leq \sigma'$.

1. Rules for Classical Propositional Logic. (α- and β-rules. See, [4].)
2. Rules for LP: (Explicit ν-rules)

EK $\dfrac{\sigma Tt{:}\varphi}{\sigma.nT\varphi}$ ($\sigma.n$ is used.) ET $\dfrac{\sigma Tt{:}\varphi}{\sigma T\varphi}$

[2] For the sake of brevity, we take $\neg\varphi$ to be an abbreviation of $\varphi \to \bot$.

E4 $\dfrac{\sigma Tt{:}\varphi}{\sigma.nTt{:}\varphi}$ ($\sigma.n$ is used.) E4r $\dfrac{\sigma.nTt{:}\varphi}{\sigma Tt{:}\varphi}$

EF $\dfrac{\sigma Ft{:}\varphi}{\sigma.nFt{:}\varphi}$ ($\sigma.n$ is used.) EFr $\dfrac{\sigma.nFt{:}\varphi}{\sigma Ft{:}\varphi}$

3. Operational Rules on F's :

!-rule $\dfrac{\sigma F!t{:}t{:}\varphi}{\sigma Ft{:}\varphi}$ ·-rule $\dfrac{\sigma F(s{\cdot}t){:}\varphi}{\sigma Ft{:}\psi\to\varphi|\sigma Fs{:}\psi}$

+-rule $\dfrac{\sigma F(s+t){:}\varphi}{\sigma Ft{:}\varphi}$ $\dfrac{\sigma F(t+s){:}\varphi}{\sigma Ft{:}\varphi}$

4. Modal Rules: π-rule: $\dfrac{\sigma F\Box\varphi}{\sigma.nF\varphi}$ ($\sigma.n$ is new.)

ν-rules: K $\dfrac{\sigma T\Box\varphi}{\sigma.nT\varphi}$ ($\sigma.n$ is used.) T $\dfrac{\sigma T\Box\varphi}{\sigma T\varphi}$ 4 $\dfrac{\sigma T\Box\varphi}{\sigma.nT\Box\varphi}$ ($\sigma.n$ is used.)

5. Constant Specification Rules: a branch is closed if it has $\sigma Fc{:}A$ (A is an axiom of S4LPN.)

We define a Fitting-style Kripke semantics for S4LPN. Let a triple (K, R, R^e) be a frame, where K is non-empty set, R is a reflexive and transitive relation on K, R^e is a reflexive, symmetric and transitive relation on K, and $R \subseteq R^e$.

Let $\mathcal{E}$ be an evidence function: $K \times Trm \longrightarrow \mathcal{P}(Fmla)$ such that

1. uR^ev implies $\mathcal{E}(u,t) \subseteq \mathcal{E}(v,t)$ (Monotonicity)[3]
2. $F \to G \in \mathcal{E}(u,t)$ and $F \in \mathcal{E}(u,s)$ implies $G \in \mathcal{E}(u, t{\cdot}s)$
3. $F \in \mathcal{E}(u,t)$ implies $t{:}F \in \mathcal{E}(u,!t)$
4. $\mathcal{E}(u,s) \cup \mathcal{E}(u,t) \subseteq \mathcal{E}(u,s+t)$
5. $\mathcal{CS} \subseteq \mathcal{E}(u,c)$.

Then, a Kripke model $\mathcal{K}$ can be defined as a quintuple $(K, R, R^e, \mathcal{E}, \mathcal{V})$. $\mathcal{V}$ is a function from propositional variables to subsets of K. We also define a forcing $\Vdash$ as a relation on $K \times Fmla$ that satisfies the following inductive property.

0. $u \Vdash p$ if and only if $u \in \mathcal{V}(p)$ and $u \nVdash \bot$ for all $u \in K$
1. $\Vdash$ commutes with propositional connectives at each state.
2. $u \Vdash \Box\varphi$ iff for every $v \in K$, s.t. uRv, $v \Vdash \varphi$
3. $u \Vdash t{:}\varphi$ iff $\varphi \in \mathcal{E}(u,t)$ and for every $v \in K$, s.t. uR^ev, $v \Vdash \varphi$.
4. $c{:}A \in \mathcal{CS}$ implies $\mathcal{K}, u \Vdash c{:}A$ for every $u \in K$.

A signed formula $F\varphi$, $T\varphi$ (written Φ schematically) is *realized* at a possible world u of a model $\mathcal{K}$ if 1) the formula is $T\varphi$ and $\mathcal{K}, u \Vdash \varphi$ or 2) the formula is $F\varphi$ and $\mathcal{K}, u \nVdash \varphi$. A set S of prefixed, signed formulas is *satisfiable* if there is a model $\mathcal{K}$ and a (partial) function $\mathcal{N}$ (called "interpretation") from the prefixes in S to possible worlds in $\mathcal{K}$, such that if $\sigma\Phi \in S$, then Φ is realized at $\mathcal{N}(\sigma)$ in $\mathcal{K}$, where Φ is a signed formula and such an $\mathcal{N}$ satisfies the condition

[3] Symmetry of uR^ev actually implies $\mathcal{E}(u,t) = \mathcal{E}(v,t)$.

(1) $\sigma \leq \sigma' \Longrightarrow \mathcal{N}(\sigma)R\mathcal{N}(\sigma')$ and (2) $1 \leq \sigma$ and $1 \leq \sigma' \Longrightarrow \mathcal{N}(\sigma)R^e\mathcal{N}(\sigma')$. A tableau branch is *satisfiable* if the set of prefixed formulas on it is satisfiable. A tableau is *satisfiable* if some tableau branch is.

3 Soundness and Completeness of the Prefixed Tableau System

We prove soundness and completeness of the prefixed tableau system.

Lemma 1. *Suppose $\mathcal{T}$ is a satisfiable tableau. If any tableau rule for* S4LPN *is applied to $\mathcal{T}$, then the resulting tableau is still satisfiable.*

Proof. Suppose a tableau is S4LPN-satisfiable because a branch θ of $\mathcal{T}$ is S4LPN-satisfiable, i.e. its members are realized at $\mathcal{N}(\sigma)$ of model $\mathcal{K}$. Suppose that a tableau rule for S4LPN is applied to the tableau $\mathcal{T}$. (We call the resulting tableau $\mathcal{T}'$.) The entire proof is divided into two cases:

Case 1: Our tableau rule is not applied on the branch θ. Then, θ is still present on the new tableau and θ is satisfiable, which makes $\mathcal{T}'$ obviously satisfiable.

Case 2: Our tableau rule is applied on the branch θ. Here we treat only relatively complicated modal cases, due to the limitation of space.

Rules for LP. Explicit ν-rules:
EFr : Suppose $\sigma.nFt:\varphi$ occurs on θ, EFr is applied and $\sigma Ft:\varphi$ is added on θ. By the assumption of the satisfiability of θ, $Ft:\varphi$ is realized at $\mathcal{N}(\sigma.n)$ in $\mathcal{K}$. So, $\mathcal{N}(\sigma.n) \nVdash t:\varphi$. Since $1 \leq \sigma.n$ and $1 \leq \sigma$, $\mathcal{N}(\sigma.n)R^e\mathcal{N}(\sigma)$ (as $\sigma.n$ is used and $\mathcal{N}$ is already defined.) By symmetry of R^e, we have $\mathcal{N}(\sigma)R^e\mathcal{N}(\sigma.n)$. By the truth condition of $t:\varphi$, (1) $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma.n), t)$ or (2) for some v, s.t. $\mathcal{N}(\sigma.n)R^ev$ and $v \nVdash \varphi$. We want to show $\mathcal{N}(\sigma) \nVdash t:\varphi$, namely $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or for some v, s.t. $\mathcal{N}(\sigma)R^ev$ and $v \nVdash \varphi$. By (1), $\mathcal{N}(\sigma)R^e\mathcal{N}(\sigma.n)$ and monotonicity, $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$. So, we can derive the desired disjunction from this. So, the first part is done. Now for (2), pick a world v such that $\mathcal{N}(\sigma.n)R^ev$ and $v \nVdash \varphi$, and name it as v_1. Since $\mathcal{N}(\sigma)R^e\mathcal{N}(\sigma.n)$, by transitivity, $\mathcal{N}(\sigma)R^ev_1$. So, for some v, $\mathcal{N}(\sigma)R^ev$ and $v \nVdash \varphi$. So, again, from this, we can derive the desired disjunction easily. Hence, $\mathcal{N}(\sigma) \nVdash t:\varphi$. So, $Ft:\varphi$ is realized at $\mathcal{N}(\sigma)$ in $\mathcal{K}$. Therefore, $\theta \cup \{\sigma Ft:\varphi\}$ is satisfiable.

Operational Rules:
!-rule: Suppose $\sigma F!t:t:\varphi$ occurs in θ, !-rule is applied on $!t:t:\varphi$ and $\sigma Ft:\varphi$ is added on θ. By the assumption of satisfiability of θ, $F!t:t:\varphi$ is realized at $\mathcal{N}(\sigma)$ in $\mathcal{K}$. So, $\mathcal{N}(\sigma) \nVdash !t:t:\varphi$. By the truth condition of $!t:t:\varphi$, (1) $t:\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), !t)$ or (2) there exists v, s. t. $\mathcal{N}(\sigma)R^ev$ and $v \nVdash t:\varphi$. Here we want to show $\mathcal{N}(\sigma) \nVdash t:\varphi$. To show this, it suffices to show $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma, t)$ or there exists v, s.t. $\mathcal{N}(\sigma)R^ev$ and $v \nVdash \varphi$. From (1), by the closure condition on $\mathcal{E}$, $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$. This suffices to derive the desired disjunction $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or there exists v, s.t. $\mathcal{N}(\sigma)R^ev$ and $v \nVdash \varphi$. So, $\mathcal{N}(\sigma) \nVdash t:\varphi$. From (2), pick some state v_1, s.t. $\mathcal{N}(\sigma)R^ev_1$ and

$v_1 \nVdash t:\varphi$. From the second part, we can obtain $\varphi \notin \mathcal{E}(v_1, t)$ or there exists v, s.t. $v_1 R^e v$ and $v \nVdash \varphi$. For the latter, we pick some state v_2, $v_1 R^e v_2$ and $v_2 \nVdash \varphi$. Since $\mathcal{N}(\sigma) R^e v_1$, by monotonicity, $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$. So, we have $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$, or $\mathcal{N}(\sigma) R^e v_1$ and $v_1 R^e v_2$ and $v_2 \nVdash \varphi$. By transitivity, the latter implies there exists v, $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \varphi$. So, $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or there exists v, $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \varphi$. Hence, $\mathcal{N}(\sigma) \nVdash t:\varphi$. So, $Ft:\varphi$ is realized at $\mathcal{N}(\sigma)$. So, $\theta \cup \{\sigma Ft:\varphi\}$ is satisfiable.

$\cdot$-rule: Suppose $\sigma Ft{\cdot}s:\varphi$ occurs on θ, $\cdot$-rule for $t{\cdot}s$ is applied and (1) $\sigma Ft:\psi \to \varphi$ is added on θ or (2) $\sigma Fs:\psi$ is added on θ (for any formula ψ). By the assumption of satisfiability of θ, $Ft{\cdot}s:\varphi$ is realized at $\mathcal{N}(\sigma)$ in some $\mathcal{K}$ under some $\mathcal{N}(\sigma)$. So, $\mathcal{N}(\sigma) \nVdash t{\cdot}s:\varphi$. By the truth condition, we have $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t{\cdot}s)$ or there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \varphi$. Here we want to show $\mathcal{N}(\sigma) \nVdash t:\psi \to \varphi$ or $\mathcal{N}(\sigma) \nVdash s:\psi$. By the closure condition of $\mathcal{E}$, $\varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t{\cdot}s)$ implies either $\psi \to \varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or $\psi \notin \mathcal{E}(\mathcal{N}(\sigma), s)$. So, the disjunction implies (1) $\psi \to \varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or (2) $\psi \notin \mathcal{E}(\mathcal{N}(\sigma), s)$ or (3) there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \varphi$.

As in the previous case, for the first two cases, we can get the respective disjunct of the goal statement; however, we need a further argument to derive the goal statement from the last part. First, note that for any formula ψ, for any $\mathcal{K}$ and for any $u \in K$, $u \Vdash \psi \vee \neg\psi$. We pick some state v_1 satisfying $\mathcal{N}(\sigma) R^e v_1$ and $v_1 \nVdash \varphi$. Since $v_1 \Vdash \psi \vee \neg\psi$, $\mathcal{N}(\sigma) R^e v_1$ and $v_1 \nVdash \varphi$ and $v_1 \Vdash \psi$ or $\mathcal{N}(\sigma) R^e v_1$ and $v_1 \nVdash \varphi$ and $v_1 \nVdash \psi$. The former implies (4) there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \psi \to \varphi$ and the latter implies (5) there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \psi$.

So, we have (1) or (2) or (4) or (5). Each of (1) and (4) implies that $\psi \to \varphi \notin \mathcal{E}(\mathcal{N}(\sigma), t)$ or there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \psi \to \varphi$. Each of (2) and (5) implies that $\psi \notin \mathcal{E}(\mathcal{N}(\sigma), s)$ or there exists v, s.t. $\mathcal{N}(\sigma) R^e v$ and $v \nVdash \psi$. Hence, $\mathcal{N}(\sigma) \nVdash t:\psi \to \varphi$ or $\mathcal{N}(\sigma) \nVdash s:\psi$. So, $Ft:\psi \to \varphi$ is realized at $\mathcal{N}(\sigma)$ or $Fs:\psi$ is realized at $\mathcal{N}(\sigma)$. Therefore, after applying π-rule for $t \cdot s$, our branch $\theta \cup \{\sigma Ft:\psi \to \varphi\}$ is satisfiable or $\theta \cup \{\sigma Fs:\psi\}$ is satisfiable.

ν-rules for $\Box$ (K, T, 4): These are the same as ordinary modal logics.
π-rule for $\Box$: The proof is similar to [4]. ⊠

Theorem 1 (Soundness).
If φ has a prefixed S4LPN-tableau proof, then φ is valid in all models.

Proof. Suppose φ has an S4LPN-tableau proof, but is not S4LPN-valid. Say, φ does not hold at world s of some S4LPN-model. Now a prefixed tableau begins with $1F\varphi$. Define an S4LPN-interpretation $\mathcal{N}$ by setting $\mathcal{N}(1) = s$. Since φ is not forced at s, i.e. $s \nVdash \varphi$, the starting S4LPN-tableau is S4LPN-satisfiable ($\mathcal{N}(1) \nVdash (\varphi)$), so $\{1F\varphi\}$ is S4LPN-satisfiable. By the lemma, so is every subsequent tableau. But an S4LPN-satisfiable tableau cannot be closed, which contradicts the assumption that φ has a tableau proof. Therefore, if φ is S4LPN-provable, then φ must be S4LPN-valid. ⊠

We move on to the completeness theorem. Our proof is done by Lindenbaum-Henkin construction in [6]. We start from some definitions. A set S of prefixed

formulas is S4LPN-consistent if no S4LPN-tableau for a finite part of S is closed. S is maximally S4LPN-consistent if S is S4LPN-consistent and no proper extension of S is S4LPN-consistent.[4] S is π-complete provided, if $\sigma\pi \in S$, then for some integer k, $\sigma.k\pi_0 \in S$ ($\sigma.kF\varphi \in S$). S omits infinitely many integers if the set of integers that do not appear in prefixes in S is infinite.

Lindenbaum-Henkin construction:
Enumerate all formulas in the language of S4LPN: $\sigma_0\Phi_0, \ldots, \sigma_n\Phi_n, \ldots$.

$$\begin{cases} S_0 = \mathrm{S}; \\ S_{n+1} = S_n \cup \{\sigma_n\Phi_n\} \text{ if this is consistent and } \Phi_n \text{ is not } \pi; \\ \qquad = S_n \cup \{\sigma_n\pi, \sigma_n.k\pi_0\} \text{ if } S_n \cup \{\sigma_n\Phi_n\} \text{ is consistent,} \\ \qquad\qquad\qquad\qquad\qquad\qquad \Phi_n \text{ is } \pi \text{ and } \sigma_n.k \text{ is new;} \\ \qquad = S_n \text{ otherwise.} \end{cases}$$

Here 'new' means that $\sigma_n.k$ does not occur in S_n or in π.

Now we state a few claims whose proofs are straightforward. (A) If S omits infinitely many integers, then this will be the case with S_n. (B) If $S_n \cup \{\sigma_n\pi\}$ is consistent, then so is $S_n \cup \{\sigma_n\pi, \sigma_n.k\pi_0\}$, provided that $\sigma_n.k$ is new. (C) If S_n omits infinitely many integers, there will always be a new prefix.

Lemma 2. *If S is consistent and omits infinitely many integers, then $\bigcup_n S_n$ ($= S_\omega$) will be maximally consistent and π-complete.*

Proof. Suppose S is consistent and omits infinitely many integers. We construct S_ω following the construction above. For maximally consistency of S_ω, the proof is essentially the same as a proof of maximal consistency of a Henkin construction for first-order logic.

Claim. S_ω is π-complete.
Proof. Suppose $\sigma F\Box\varphi \in S_\omega$. Then, there exists n such that $\sigma = \sigma_n$ and $F\Box\varphi = \Phi_n$ and $S_n \cup \{\sigma_n\Phi_n\}$ is consistent. (Otherwise, $\sigma F\Box\varphi$ would not be in S_ω.) By construction, $\sigma_n.kF\varphi$ has to be in S_{n+1}. Indeed, we can show the following. By assumption, S omits infinitely many integers. So, S_{n+1} omits infinitely many integers (by claim (A)). Hence, by claim (C), it is always possible to find a new k, s.t. $\sigma_n.kF\varphi \in S_{n+1} \subseteq S_\omega$, as desired. Therefore, S_ω is π-complete. ⊠(claim)

These suffice to show the lemma. ⊠(lemma)

We construct a canonical Kripke model $\mathcal{K} = (K, R, R^e, \mathcal{E}, \mathcal{V})$ for S4LPN based on this maximal consistent set. Let K be the set of prefixes that occur in S_ω. Possible worlds will be taken to be syntactic objects, prefixes, just as in the usual Henkin construction objects in the domain are syntactic objects, terms in the language. The accessibility relations R, R^e, propositional valuation $\mathcal{V}$ and evidence function $\mathcal{E}$ are given as follows: 1. $\sigma R\sigma'$ iff σ is an initial segment of σ' ($\sigma \leq \sigma'$); 2. $\sigma R^e\sigma'$ iff $1 \leq \sigma$ and $1 \leq \sigma'$; 3. $\sigma \in \mathcal{V}(p)$ iff $\sigma Tp \in S_\omega$; 4. $\varphi \in \mathcal{E}(\sigma, t)$ iff $\sigma Ft{:}\varphi \notin S_\omega$.

[4] In the following, we use "consistent" to mean "S4LPN-consistent."

Now we check that $\mathcal{E}$, R and R^e defined in the canonical model satisfy the conditions of a model of S4LPN. However, by construction of our canonical model, it is obvious that R is a reflexive and transitive relation and R^e is an equivalence relation. So, we focus on the conditions of $\mathcal{E}$. We first prove a useful proposition.

Proposition 1.

1. $\sigma Ft{:}\varphi \in S_\omega$ *if and only if* $\sigma' Ft{:}\varphi \in S_\omega$ (*for any* $1 \leq \sigma$ *and* $1 \leq \sigma'$)
2. $\sigma Tt{:}\varphi \in S_\omega$ *if and only if* $\sigma' Tt{:}\varphi \in S_\omega$ (*for any* $1 \leq \sigma$ *and* $1 \leq \sigma'$)

Proof. 1. Suppose $\sigma Ft{:}\varphi \in S_\omega$ but $\sigma' Ft{:}\varphi \notin S_\omega$. Then, by maximal consistency, $S_\omega \cup \{\sigma' Ft{:}\varphi\}$ is inconsistent, namely there is a finite subset S_1 of $S_\omega \cup \{\sigma' Ft{:}\varphi\}$ that has a closed tableau. However, if so, we can produce another closed tableau by putting $\sigma Ft{:}\varphi$ on top of it and applying EF_r and EF finitely many times. (First get $1Ft{:}\varphi$ from $\sigma Ft{:}\varphi$ by using EF_r and then get $\sigma' Ft{:}\varphi$ by using EF from $1Ft{:}\varphi$.) This new closed tableau can be taken to be a closed tableau for $(S_1 \backslash \{\sigma' Ft : \varphi\}) \cup \{\sigma Ft : \varphi\}$, since everywhere $\sigma' Ft : \varphi$ is used, we can replace them by the figure given above from $\sigma Ft{:}\varphi$. So, this set has a closed tableau. Note that $(S_1 \backslash \{\sigma' Ft{:}\varphi\}) \cup \{\sigma Ft{:}\varphi\} \subseteq S_\omega$. This implies that S_ω is inconsistent. Contradiction. The proof of the converse is similar. The proof of 2 is also similar but with E4 and E4r. ⊠

Proposition 2. *The evidence function defined above satisfies the following conditions: (1) monotonicity, (2) closure conditions, (3) constant specification.*

Proof. (1) (Monotonicity) Suppose $\sigma R^e \sigma'$ and $\varphi \in \mathcal{E}(\sigma, t)$. By definition, $\sigma Ft : \varphi \notin S_\omega$. So, proposition 1, $\sigma' Ft{:}\varphi \notin S_\omega$. So, $\varphi \in \mathcal{E}(\sigma', t)$.

(2) (Closure Conditions 2.) Suppose $\varphi \notin \mathcal{E}(\sigma, t{\cdot}s)$. By definition, $\sigma Ft{\cdot}s{:}\varphi \in S_\omega$. We consider applying $t{\cdot}s$-rule to a finite subset of S_ω.

Claim. $\sigma Ft{:}\psi \to \varphi \in S_\omega$ or $\sigma Fs{:}\psi \in S_\omega$.

Proof. First, we show that $S_\omega \cup \{\sigma Ft{:}\psi \to \varphi\}$ is consistent or $S_\omega \cup \{\sigma Fs{:}\varphi\}$ is consistent. Suppose otherwise, i.e., $S_\omega \cup \{\sigma Ft : \psi \to \varphi\}$ is inconsistent and $S_\omega \cup \{\sigma Fs : \varphi\}$ is inconsistent. Then, there exists a finite set S^1 s.t. $S^1 \subseteq S_\omega \cup \{\sigma Ft{:}\psi \to \varphi\}$ and $\sigma Ft{:}\psi \to \varphi \in S^1$ and S^1 has a closed tableau and there exists a finite set S^2 s.t. $S^2 \subseteq S_\omega \cup \{\sigma Fs{:}\psi\}$, $\sigma Fs{:}\psi \in S^2$ and S^2 has a closed tableau. (Note that since S_ω is consistent, we have to use an additional formula to close a tableau for each case.) These formulas are obtained by applying $\cdot$-rule. So, we can construct another closed tableau for a finite set $S^1 \cup S^2 \cup \{\sigma Ft{\cdot}s{:}\varphi\}$ by taking the tableau for S^1 and S^2 as branches of the new tableau and by applying $\cdot$-rule. However, $S^1 \subseteq S_\omega$, $S^2 \subseteq S_\omega$ and $\{\sigma Ft{\cdot}s{:}\varphi\} \subseteq S_\omega$. So, a finite set of S_ω has a closed tableau, which contradicts the consistency of S_ω. By maximality, $\sigma Ft{:}\psi \to \varphi \in S_\omega$ or $\sigma Fs{:}\psi \in S_\omega$. ⊠

By definition, $\psi \to \varphi \notin \mathcal{E}(\sigma, t)$ or $\psi \notin \mathcal{E}(\sigma, s)$.

(Closure Condition 3.) Suppose $t{:}\varphi \notin \mathcal{E}(\sigma, !t)$. By definition, $\sigma F!t{:}t{:}\varphi \in S_\omega$. We want to show $\sigma Ft{:}\varphi \in S_\omega$. To show that, it suffices to show the consistency of

$S_\omega \cup \{\sigma Ft{:}\varphi\}$, due to maximal consistency of S_ω. Suppose otherwise, i.e. this set is inconsistent. Then, there is a finite subset of this set that has a closed tableau. Pick up one such closed tableau. Put $\sigma F!t{:}t{:}\varphi$ on top of it and apply !-rule, then we can produce another closed tableau of a finite subset of $S_\omega \cup \{\sigma F!t : t : \varphi\}$. However, $\sigma F!t : t : \varphi \in S_\omega$. This implies S_ω is inconsistent. Contradiction. So, $S_\omega \cup \{\sigma Ft : \varphi\}$ is consistent and, by maximality of S_ω, $\sigma Ft : \varphi \in S_\omega$. So, by definition, $\varphi \notin \mathcal{E}(\sigma, t)$.

Proofs for (Closure Condition 4.) and (3) Constant Specification are similar. ⊠

Lemma 3 (Truth Lemma). *For every signed formula Φ, $\sigma\Phi \in S_\omega \Longrightarrow \sigma$ realizes Φ in the model $\mathcal{K}$.*

Proof. By induction on complexity of formulas. We show $\Box$ and $t{:}$cases.

Case 1. $\Phi = T\Box\varphi$. Suppose $\sigma T\Box\varphi \in S_\omega$. By the tableau rule K and 4, $S_\omega \cup \{\sigma.nT\varphi, \sigma.nT\Box\varphi\}$ is consistent for any $\sigma.n$ occurring in S_ω. So, by maximality, $\sigma.nT\varphi, \sigma.nT\Box\varphi \in S_\omega$, for any $\sigma.n$ occurring in S_ω. By using this combination finitely many times, we can show that for any arbitrary sequence σ', s.t. $\sigma \leq \sigma'$, $\sigma' T\varphi \in S_\omega$. By IH, σ' realizes $T\varphi$ in the model $\mathcal{K}$. So, $\mathcal{K}, \sigma' \Vdash \varphi$. So, for any $\sigma' \geq \sigma$, $\mathcal{K}, \sigma' \Vdash \varphi$. So $\mathcal{K}, \sigma \Vdash \Box\varphi$. Hence, σ realizes $T\Box\varphi$ in the model $\mathcal{K}$.

Case 2. $\Phi = F\Box\varphi$. Suppose $\sigma F\Box\varphi \in S_\omega$. By π-completeness of S_ω, $\sigma.kF\varphi \in S_\omega$ for some $\sigma.k$ occurring in S_ω. By IH, $\sigma.k$ realizes $F\varphi$ in the model $\mathcal{K}$. So, $\mathcal{K}, \sigma.k \nVdash \varphi$. On the other hand, clearly, $\sigma \leq \sigma.k$ So, there exists $\sigma.k \geq \sigma$, $\mathcal{K}, \sigma.k \nVdash \varphi$. So $\mathcal{K}, \sigma \nVdash \Box\varphi$. Hence, σ realizes $F\Box\varphi$ in the model $\mathcal{K}$.

Case 3. $\Phi = Tt{:}\varphi$. Suppose $\sigma Tt{:}\varphi \in S_\omega$. It suffices to show that $\sigma \Vdash t{:}\varphi$. To show this, it is suffices to show the two statements: (1) $\varphi \in \mathcal{E}(\sigma, t)$ and (2) for all σ', $(\sigma R^e \sigma' \Rightarrow \sigma' \Vdash \varphi)$. (1) is an immediate consequence of the definition of $\mathcal{E}$ and $\sigma Ft{:}\varphi \notin S_\omega$ (by consistency). So, $\varphi \in \mathcal{E}(\sigma, t)$. To show (2), suppose $\sigma R^e \sigma'$. By definition, $1 \leq \sigma$ and $1 \leq \sigma'$. Since σ and σ' are related in this way, by the proposition 1 combined with the rule ET, $\sigma Tt{:}\varphi \in S_\omega$ implies $\sigma' T\varphi \in S_\omega$. By IH, σ' realizes φ in $\mathcal{K}$. So, $\sigma' \Vdash \varphi$. So, for all σ', s.t. $\sigma R^e \sigma'$, $\sigma' \Vdash \varphi$. Therefore, $\sigma \Vdash t{:}\varphi$. So, σ realizes $t{:}\varphi$ in $\mathcal{K}$.

Case 4. $\Phi = Ft : \varphi$. Suppose $\sigma Ft : \varphi \in S_\omega$. It suffices to show $\sigma \nVdash t : \varphi$. To show this, it is sufficient to show (1) $\varphi \notin \mathcal{E}(\sigma, t)$ or (2) there exists σ', s.t. $\sigma R^e \sigma'$ and $\sigma' \nVdash \varphi$. By the assumption, it is not the case that $\sigma Ft : \varphi \notin S_\omega$. So, this immediately implies $\varphi \notin \mathcal{E}(\sigma, t)$. So, $\varphi \notin \mathcal{E}(\sigma, t)$ or there exists σ', s.t. $\sigma R^e \sigma'$ and $\sigma' \nVdash \varphi$. So, $\sigma \Vdash t{:}\varphi$. Hence, σ realizes $Ft{:}\varphi$ in $\mathcal{K}$. ⊠

Theorem 2 (Completeness). *If φ is S4LPN-valid, then φ has a proof using the tableau rules for S4LPN.*

Proof. We show the contrapositive. Suppose φ is not provable using the prefixed S4LPN-tableau rules. Then $\{1F\varphi\}$ is S4LPN-consistent, and it omits infinitely many integers. So, we can extend it to a maximally S4LPN-consistent, π-complete set S_ω by the above construction and the lemma 6. We can define a canonical Kripke model $\mathcal{K}$ out of S_ω. By Truth Lemma, we can show $F\varphi$ is realized at 1 in $\mathcal{K}$. So, there is a Kripke model $\mathcal{K}$ and a state σ such that $\mathcal{K}, \sigma \nVdash \varphi$. ⊠

4 Hypersequent Calculus for S4LPN

Here we first present the hypersequent calculus HS4LPN. And then we give a translation from the prefixed tableau system to a hypersequent calculus of S4LPN. A hypersequent is a multiset of sequents, written as follows: $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_n \Rightarrow \Delta_n$.

1) **Axiom**: $A \Rightarrow A$ $\qquad$ $\bot \Rightarrow$

2) **External structural rules**:[5]

$$\mathbf{EW}\ \frac{G}{G|H} \qquad \mathbf{EC}\ \frac{G|\Gamma \Rightarrow \Delta|\Gamma \Rightarrow \Delta}{G|\Gamma \Rightarrow \Delta}$$

3) **Internal structural rules**:

$$\mathbf{LW}\ \frac{G|\Gamma \Rightarrow \Delta}{G|A, \Gamma \Rightarrow \Delta} \qquad \mathbf{RW}\ \frac{G|\Gamma \Rightarrow \Delta}{G|\Gamma \Rightarrow \Delta, A}$$

$$\mathbf{LC}\ \frac{G|A, A, \Gamma \Rightarrow \Delta}{G|A, \Gamma \Rightarrow \Delta} \qquad \mathbf{RC}\ \frac{G|\Gamma \Rightarrow \Delta, A, A}{G|\Gamma \Rightarrow \Delta, A}$$

4) **Logical rules**:

$$\mathbf{L}\wedge\ \frac{G|A, B, \Gamma \Rightarrow \Delta}{G|A \wedge B, \Gamma \Rightarrow \Delta} \qquad \mathbf{R}\wedge\ \frac{G|\Gamma \Rightarrow \Delta, A \qquad G|\Gamma \Rightarrow \Delta, B}{G|\Gamma \Rightarrow \Delta, A \wedge B}$$

$$\mathbf{L}\vee\ \frac{G|A, \Gamma \Rightarrow \Delta \quad G|B, \Gamma \Rightarrow \Delta}{G|A \vee B, \Gamma \Rightarrow \Delta} \qquad \mathbf{R}\vee\ \frac{G|\Gamma \Rightarrow \Delta, A, B}{G|\Gamma \Rightarrow \Delta, A \vee B}$$

$$\mathbf{L}\rightarrow\ \frac{G|\Gamma \Rightarrow \Delta, A \qquad G|B, \Gamma \Rightarrow \Delta}{G|A \rightarrow B, \Gamma \Rightarrow \Delta} \qquad \mathbf{R}\rightarrow\ \frac{G|A, \Gamma \Rightarrow \Delta, B}{G|\Gamma \Rightarrow \Delta, A \rightarrow B}$$

5) **Rules for S4$\Box$**: $\mathbf{L}\Box\ \dfrac{G|A, \Gamma \Rightarrow \Delta}{G|\Box A, \Gamma \Rightarrow \Delta}$ $\qquad$ $\mathbf{R}\Box\ \dfrac{G|\Box\Gamma \Rightarrow A}{G|\Box\Gamma \Rightarrow \Box A}$

6) **Rules for Proof-terms of LP**:

$$\mathbf{Lt}\ \frac{G|A, \Gamma \Rightarrow \Delta}{G|t{:}A, \Gamma \Rightarrow \Delta} \qquad \mathbf{R}\cdot\ \frac{G|\Gamma \Rightarrow \Delta, t{:}A \rightarrow B \qquad G|\Gamma \Rightarrow \Delta, s{:}A}{G|\Gamma \Rightarrow \Delta, t{\cdot}s{:}B}$$

$$\mathbf{R}!\ \frac{G|\Gamma \Rightarrow \Delta, t{:}A}{G|\Gamma \Rightarrow \Delta, !t{:}t{:}A} \qquad \mathbf{R}+\ \frac{G|\Gamma \Rightarrow \Delta, t{:}A}{G|\Gamma \Rightarrow \Delta, t+s{:}A} \qquad \frac{G|\Gamma \Rightarrow \Delta, s{:}A}{G|\Gamma \Rightarrow \Delta, t+s{:}A}$$

7) **Constant Specification**: $\dfrac{}{\Rightarrow c{:}A}$ where A is an axiom of S4LPN.

8) **Labeled Splitting**[6]: $\dfrac{G|\boldsymbol{t}{:}\Gamma_1, \Gamma_2 \Rightarrow \boldsymbol{s}{:}\Delta_1, \Delta_2}{G|\boldsymbol{t}{:}\Gamma_1 \Rightarrow \boldsymbol{s}{:}\Delta_1 | \Gamma_2 \Rightarrow \Delta_2}$

[5] Since we are assuming that all the sequents and hypersequents in our system are multi-sets, so we do not need any exchange rules, internal or external.

[6] Here $\boldsymbol{t}{:}\Gamma_1 = t_1{:}\varphi_1, \ldots, t_n{:}\varphi_n$. The rule covers cases where $\boldsymbol{t}{:}\Gamma_1$ or $\boldsymbol{s}{:}\Delta_1$ is empty ([3]). Note also that this rule has its origin in Avron's modal splitting rule in his hypersequent calculus for S5, which can handle the negative introspection in S5.

9) **Cut**: $$\frac{G_1|\Gamma_1 \Rightarrow \Delta_1, A \qquad G_2|A, \Gamma_2 \Rightarrow \Delta_2}{G_1|G_2|\Gamma_1, \Gamma_2 \Rightarrow \Delta_1, \Delta_2}$$

E.g., a derivation in HS4LPN.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{t{:}A \Rightarrow t{:}A}{t{:}A \Rightarrow !t{:}t{:}A}}{t{:}A \Rightarrow \mid \Rightarrow !t{:}t{:}A}}{\Rightarrow \neg t{:}A|\neg !t{:}t{:}A \Rightarrow}}{\Rightarrow \Box\neg t{:}A|\neg !t{:}t{:}A \Rightarrow}}{\neg !t{:}t{:}A \Rightarrow \Box\neg t{:}A|\neg !t{:}t{:}A \Rightarrow \Box\neg t{:}A}}{\neg !t{:}t{:}A \Rightarrow \Box\neg t{:}A}$$

a few logical rules

a few IW

By the following translation from hypersequents to formulas in the language of S4LPN, we can prove that the Hilbert-style system S4LPN and HS4LPN are deductively equivalent.

$$\mathcal{I}(\Gamma_1 \Rightarrow \Delta_1|\ldots|\Gamma_n \Rightarrow \Delta_n) = \Box(\bigwedge \Gamma_1 \to \bigvee \Delta_1) \vee \cdots \vee \Box(\bigwedge \Gamma_n \to \bigvee \Delta_n)$$

Theorem 3. *HS4LPN* $\vdash \Gamma_1 \Rightarrow \Delta_1|\ldots|\Gamma_n \Rightarrow \Delta_n$ *if and only if S4LPN* $\vdash \Box(\bigwedge \Gamma_1 \to \bigvee \Delta_1) \vee \cdots \vee \Box(\bigwedge \Gamma_n \to \bigvee \Delta_n)$.

Proof. In both ways, a proof is given by induction on the length of derivation. To derive Labeled Splitting (from right to left), we can use the fact S4LPN $\vdash (t_1{:}A \to t_2{:}B) \to \Box(t_1{:}A \to t_2{:}B)$. ⊠

However, to show that Cut is admissible, we need another argument. Here we give a semantic proof of cut-admissibility. Since the Hilbert-style system S4LPN is sound and HS4LPN is equivalent to that system, HS4LPN is sound with respect to the Kripke semantics given above. So, to show cut-admissibility, it suffices to show completeness of HS4LPN without Cut (HS4LPN$^-$) with respect to the same semantics. We show completeness of HS4LPN$^-$ by providing a way of translating a prefixed tableau proof to a proof in the hypersequent calculus, following [6].

We define a translation from the language of the prefixed tableau system to that of the hypersequent calculus. Our translation mapping, which is called s, is defined in two stages. First, we define a mapping t that maps a set of prefixed formulas to a multi-set of multi-sets of signed formulas in the following way. 1. The set of prefixed formulas is partitioned into subsets so that all formulas with the same prefixes σ_i go into the same subset. 2. We strip off prefixes from those partitioned prefixed formulas (for each σ_i). 3. For each σ_i, we put together those signed formulas and the formulas of the form $T\Box\psi$ such that $\sigma_j T\Box\psi \in S$ and $\sigma_j < \sigma_i$ for some σ_j. (The prefix of $T\Box\psi$ is a proper initial segment of σ_i.) 4. We call the resulting set H_{σ_i} for each σ_i, i.e. $H_{\sigma_i} := \{\Phi|\sigma_i\Phi \in S\} \cup \{T\Box\varphi|\exists\sigma_j < \sigma_i(\sigma_j T\Box\varphi \in S)\}$. 5. H_{σ_i}'s get linearly ordered by using the lexicographic ordering induced by the partial order of σ_i's.6. We arrange those linearly ordered H_{σ_i} by using "|" in the hypersequent calculus. So, we have $H_{\sigma_1}|H_{\sigma_2}|\ldots|H_{\sigma_n}$. Our reading "|" is the same as that of

hypersequent, so we have now constructed a multi-set of sets of signed formulas.[7] Let $S^t := H_{\sigma_1}|H_{\sigma_2}|\dots|H_{\sigma_n}$.

Second, we consider a mapping that maps a multi-set of multi-sets of signed formulas, i.e. $H_{\sigma_1}|H_{\sigma_2}|\dots|H_{\sigma_n}$, to a multi-set of sequents. This mapping can be constructed by putting T formulas to the antecedent and F formulas to the succedent for each case of H_{σ_i}. Namely, if $H_{\sigma_i} = \{T\varphi_1, \dots, T\varphi_k, F\psi_1, \dots, F\psi_m\}$, then we map this to $\varphi_1, \dots, \varphi_k \Rightarrow \psi_1, \dots, \psi_m$. This is all we have to do for each H_{σ_i}, but, for simplicity, we officially define mapping for an entire multi-set of H_{σ_i}'s. We call this mapping u. So, we have $(H_{\sigma_1}|H_{\sigma_2}|\dots|H_{\sigma_n})^u = \varphi_1, \dots, \varphi_{k_{\sigma_1}} \Rightarrow \psi_1, \dots, \psi_{m_{\sigma_1}}|\dots|\varphi_1, \dots, \varphi_{k_{\sigma_n}} \Rightarrow \psi_1, \dots, \psi_{m_{\sigma_n}}$.

We finally define the desired mapping s as a composition of t and u, i.e., $S^s := ((S^t)^u)$. So, S^s will be of the form $\Gamma_1 \Rightarrow \Delta_1|\dots|\Gamma_{\sigma_i} \Rightarrow \Delta_{\sigma_i}|\dots|\Gamma_{\sigma_n} \Rightarrow \Delta_{\sigma_n}$.[8]

Lemma 4. *Let S be a finite set of prefixed signed formulas (only with prefixes that have the initial element 1). If there is a closed tableau for S using the prefixed tableau system for* S4LPN*, then the hypersequent S^s is provable in* HS4LPN$^-$*.*

Proof. Proof by induction on the depth d of the tableau proof, where the depth is the least number d such that there is a closed prefixed tableau in S4LPN for S with d applications of tableau rules. Suppose a tableau for S closes with depth d and, by IH, the theorem holds for sets that close with less than d. Suppose we have made the first application of a tableau rule. We call the resulting set S'. Then, to obtain a closed tableau, we only need $d-1$ applications of the rules. So, for any tableau rule, after this first application we can apply IH. Namely, by IH, if there is a closed tableau for S' by $d-1$ applications of rules, then the hypersequent $(S')^s$ is provable in HS4LPN$^-$. So, what is remaining is to show how we can prove that hypersequent S^s which is obtained by mapping the set S we had before we apply the rule, depending on which rule is applied as the first step of a tableau proof.

Base Case: $d = 0$, i.e. the original set S itself is trivially a closed tableau. Let $S = \{1T\psi_1, \dots, \sigma_n T\psi_n, \sigma T\varphi, \sigma F\varphi, 1F\rho_1, \dots, \sigma_m F\rho_m\}$. Then, $S^s = \psi_1 \Rightarrow \rho_1|\dots|\psi_n \Rightarrow |\dots|\varphi \Rightarrow \varphi|\dots| \Rightarrow \rho_m$. This hypersequent is provable in HS4LPN$^-$ by an axiom $\varphi \Rightarrow \varphi$ and applying IW and EW.

Inductive Cases: (We present representative cases of tableau rules.)[9]

[7] Each H_{σ_i} will work as each sequent occurring in the hypersequent that we will obtain as an image of the mapping s defined here. But each H_{σ_i} consists of only signed formulas, so prefixes stop playing a mathematical role, once we are done with the above arrangement.

[8] In the following, we suppress the modal formulas added to sequents by our mapping, unless we need to explicitly write them. Also, note that the original S is a set of prefixed formulas and S^s is a multi-set of multi-sets, but due to the structural rules in our hypersequent calculus, this difference does not produce any defect in preserving derivability.

[9] Except for the cases where we explicitly note, we adopt the notational convention that for the original Γ_σ (or Δ_σ), we write the formula at issue explicitly. So, e.g., our "Γ_σ" is really $\Gamma_\sigma \backslash \{t{:}\varphi\}$. Also, we suppress double lines even when several steps are taken.

Case 1. EFr.

Suppose $\sigma.nFt:\varphi \in S$. By the assumption, such an S has a closed tableau. Also, since the first application of a rule in the tableau is EFr, $S \cup \{\sigma Ft:\varphi\}$ has a closed tableau with depth $d-1$. By IH, $(S \cup \{\sigma Ft:\varphi\})^s$ is provable in $\mathsf{HS4LPN}^-$, where this has the form $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, t:\varphi | \ldots | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}$. We show S^s, which is equal to $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}$, is provable only by using rules of $\mathsf{HS4LPN}^-$.

$$\dfrac{\dfrac{\dfrac{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, t:\varphi | \ldots | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma | \Rightarrow t:\varphi | \ldots | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}}{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma | \ldots | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}}{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma | \ldots | \Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}, t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}$$

Case 2. !.

Suppose $\sigma F!t:t:\varphi \in S$. By the assumption, such an S has a closed tableau. Also, the first application of a rule in the tableau is !-rule, and $S \cup \{\sigma Ft:\varphi\}$ has a closed tableau with depth $d-1$. By IH, $(S \cup \{\sigma Ft:\varphi\})^s$ is provable in $\mathsf{HS4LPN}^-$, i.e. $\mathsf{HS4LPN}^- \vdash \Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, t:\varphi, !t:t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}$. The following proof is enough to show the provability of S^s in $\mathsf{HS4LPN}^-$.

$$\dfrac{\dfrac{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, t:\varphi, !t:t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, !t:t:\varphi, !t:t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}}{\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma_\sigma \Rightarrow \Delta_\sigma, !t:t:\varphi | \ldots | \Gamma_{\sigma_m} \Rightarrow \Delta_{\sigma_m}}$$

Case 3. K. Suppose $\sigma T\Box\varphi \in S$. By the assumption, such an S has a closed tableau with depth d. Since the first application of a rule in this tableau is K-rule, $S \cup \{\sigma.nT\varphi\}$ has a closed tableau with depth $d-1$. By IH, $(S \cup \{\sigma.nT\varphi\})^s$ provable in $\mathsf{HS4LPN}^-$. By definition of s, for any $\sigma' \geq \sigma$ occurring in $S \cup \{\sigma.nT\varphi\}$, $\Box\varphi \in \Gamma_{\sigma'}$. Since $\sigma \leq \sigma.n$, in particular, $\Box\varphi \in \Gamma_\sigma$ and $\Box\varphi, \varphi \in \Gamma_{\sigma.n}$. Let us write down Γ_σ as $\Gamma'_\sigma, \Box\varphi$ and $\Gamma_{\sigma.n}$ as $\Gamma'_{\sigma.n}, \Box\varphi, \varphi$. So, the image of the set given above by s has the form $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma'_\sigma, \Box\varphi \Rightarrow \Delta_\sigma | \ldots | \Gamma'_{\sigma.n}, \Box\varphi, \varphi \Rightarrow \Delta_{\sigma.n} | \ldots | \Gamma_{\sigma_m}, \Box\varphi \Rightarrow \Delta_{\sigma_m}$.[10] We show below that S^s, which is equal to $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma'_\sigma, \Box\varphi \Rightarrow \Delta_\sigma | \Gamma'_{\sigma.n}, \Box\varphi \Rightarrow \Delta_{\sigma.n} | \ldots | \Gamma_{\sigma_m}, \Box\varphi \Rightarrow \Delta_{\sigma_m}$, is provable in $\mathsf{HS4LPN}^-$.

$$\dfrac{\dfrac{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\varphi \Rightarrow \Delta_\sigma | \ldots | \Gamma'_{\sigma.n}, \Box\varphi, \varphi \Rightarrow \Delta_{\sigma.n} | \ldots | \Gamma_{\sigma_m}, \Box\varphi \Rightarrow \Delta_{\sigma_m}}{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\varphi \Rightarrow \Delta_\sigma | \ldots | \Gamma'_{\sigma.n}, \Box\varphi, \Box\varphi \Rightarrow \Delta_{\sigma.n} | \ldots | \Gamma_{\sigma_m}, \Box\varphi \Rightarrow \Delta_{\sigma_m}}}{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\varphi \Rightarrow \Delta_\sigma | \ldots | \Gamma'_{\sigma.n}, \Box\varphi \Rightarrow \Delta_{\sigma.n} | \ldots | \Gamma_{\sigma_m}, \Box\varphi \Rightarrow \Delta_{\sigma_m}}$$

Case 4. π-rule

Suppose $\sigma F\Box\varphi \in S$. By the assumption, such an S has a closed tableau with depth d. Since the first application of a rule in this tableau is π-rule, $S \cup \{\sigma.nF\varphi\}$ (n is new) has a closed tableau with depth $d-1$.

By IH, we have the provability of $(S \cup \{\sigma.nF\varphi\})^s$ ($\sigma F\Box\varphi \in S$) in the hypersequent calculus $\mathsf{HS4LPN}^-$. We have to consider a general case in which

[10] Note that for σ_m, we have extra $\Box\varphi$ if $\sigma \leq \sigma_m$, due to our definition of mapping s. We suppress other sequents. In the following, we always assume σ_m to be $\sigma \leq \sigma_m$.

$S \cup \{\sigma.nF\varphi\}$ contains some $\sigma T\Box\varphi$. So, let us assume $\Box\varphi_i \in \Gamma_\sigma$ $(1 \leq i \leq k)$. Only here we use a notation $\Gamma_\sigma := \Gamma'_\sigma, \Box\varphi_1, \ldots, \Box\varphi_k$. By definition of s and $\sigma \leq \sigma.n$, for the prefix newly introduced when π-rule is applied, we have $\Gamma_{\sigma.n} = \Box\varphi_1, \ldots, \Box\varphi_k = \{\Box\varphi | \Box\varphi \in \Gamma_\sigma\}$. (Let us put $\Gamma_{\sigma.n} = \Box\varphi_1, \ldots, \Box\varphi_k := \Box\Gamma''$).[11] Also, we have $\Delta_{\sigma.n} = \{\varphi\}$ and let us use the notation $\Delta_\sigma := \Delta'_\sigma, \Box\varphi$. So, the above set has the form $\Gamma_1 \Rightarrow \Delta_1 | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Box\Gamma'' \Rightarrow \varphi | \ldots | \Gamma_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}$.

S^s is $\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Gamma_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}$.

Note that $\sigma.n$ is new. One sequent in the hypersequent $\Gamma_{\sigma.n} \Rightarrow \Delta_{\sigma.n}$ disappears when we reach the conclusion of the derivation in $\mathsf{HS4LPN}^-$, and this is what it should be since the original set S for the prefixed tableau system did not have formulas corresponding to that. We can emulate this feature of the prefixed tableau system in $\mathsf{HS4LPN}^-$ by EC.

$$\dfrac{\dfrac{\dfrac{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Box\Gamma'' \Rightarrow \varphi | \ldots | \Gamma_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}}{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Box\Gamma'' \Rightarrow \Box\varphi | \ldots | \Gamma_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}}}{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Gamma'_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}}}{\Gamma_{\sigma_1} \Rightarrow \Delta_{\sigma_1} | \ldots | \Gamma'_\sigma, \Box\Gamma'' \Rightarrow \Delta'_\sigma, \Box\varphi | \ldots | \Gamma_{\sigma_m}, \Box\Gamma'' \Rightarrow \Delta_{\sigma_m}}$$

This derivation shows the provability of S^s. ⊠

Theorem 4 (Completeness). *If a* formula φ *is valid in the semantics for* S4LPN*, then* $\mathsf{HS4LPN}^- \vdash\!\Rightarrow \varphi$.

Proof. Use completeness of the prefixed tableau system. Then, apply Lemma 4. This is a special case of the lemma where $S = \{1F\varphi\}$. ⊠

As a corollary, we can semantically prove cut-admissibility for HS4LPN.

Corollary 1. *If* $\mathsf{HS4LPN} \vdash\!\Rightarrow \varphi$*, then* $\mathsf{HS4LPN}^- \vdash\!\Rightarrow \varphi$

Remark 1. In spite of its cut-admissibility, the hypersequent calculus HS4LPN is not analytic, i.e. does not enjoy the subformula property, due to the rule $\mathbf{R}\cdot$. So, the hypersequent calculus may not be useful for automated reasoning in the logic. We leave this issue for future research.

References

1. Artemov, S.N.: Explicit Provability and Constructive Semantics. The Bulletin of Symbolic Logic 7(1), 1–36 (2000)
2. Artemov, S.N., Nogina, E.: Introducing Justification into Epistemic Logic. J. Log. Comput. 15(6) (2005)

[11] Note that $\Box\Gamma'_{\sigma.n}$ can contain only $\Box$-ed formulas that come from Γ_σ, since $\sigma.n$ is an immediate successor of σ and that all the $T\Box$ formulas whose prefix τ is such that $\tau \leq \sigma$ must already be in Γ_σ by the definition of mapping applied to $S \cup \{\sigma.nF\varphi\}$. Also, Γ_{σ_m} must contain $\Box\Gamma''$, since the default assumption here is $\sigma \leq \sigma_m$.

3. Avron, A.: The Method of Hypersequents in the Proof Theory of Propositional Non-classical Logics. In: Hodges, W., Hyland, M., Steinhorn, C., Truss, J. (eds.) Logic: from foundations to applications. Proc. Logic Colloquium, Keele, UK, 1993, pp. 1–32. Oxford University Press, New York (1996)
4. Fitting, M.: Proof Methods for Modal and Intuitionistic Logic. Reidel Publishing Company (1983)
5. Fitting, M.: Semantics and Tableaus for LPS4. Technical report, CUNY Ph.D. Program in Computer Science Technical Report TR-2004016 (2004)
6. Fitting, M.: Modal Proof Theory. In: Handbook of Modal Logic. Elsevier, New York (2006)
7. Gödel, K.: Vortrag bei Zilsel (1938). In: Feferman, S., et al. (eds.) Kurt Gödel, Collected Works, vol. III. Oxford Univ. Press, New York (1995)
8. Nogina, E.: Epistemic Completeness of GLA. The Bulletin of Symbolic Logic 13(3), 407 (2007)
9. Renne, B.: Semantic Cut-Elimination for Two Explicit Modal Logics. In: Huitnink, J., Katrenko, S. (eds.) Proceedings of the 11th ESSLLI Student Session, Málaga, Spain, pp. 148–158 (2006)
10. Yavorskaya, T.: Logic of Proofs and Provability. Ann. Pure Appl. Logic 113(1-3) (2001)

Topological Forcing Semantics with Settling

Robert S. Lubarsky

Department of Mathematical Sciences
Florida Atlantic University
777 Glades Rd.
Boca Raton, FL 33431
rlubarsk@fau.edu

Abstract. Just as forcing, or Boolean-valued models, produce a model of ZF (when the meta-theory is, or the ground model satisfies, ZF), so do Heyting-valued models satisfy IZF, which stands for Intuitionistic ZF, the standard constructive re-working of the ZF axioms. In this paper, a variant model is introduced (with truth values the Heyting algebra of open sets of a topological space), along with a correspondingly revised forcing or satisfaction relation. Such a model is shown to satisfy only a fragment of IZF. Natural properties of the underlying topological space are shown to imply stronger closure properties of the model. (It is impossible, except in trivial cases, for Power Set to be satisfied.) This semantics generalizes the second model of [9], which is the current semantics for the special case of the underlying topological space being $\mathbb{R}$.

Keywords: Constructivism, set theory, semantics, topology AMS classification 03F50, 03E70, 03C90.

1 Introduction

Topological interpretations of constructive systems were first studied by Stone [15] and Tarski [16], who independently provided such for propositional logic. This was later extended by Mostowski [12] to predicate logic. The first application of this to any sort of higher-order system was Scott's interpretation of analysis [13,14]. Grayson [6,7] then generalized the latter to the whole set-theoretic universe, to provide a model of IZF, Intuitionistic Zermelo-Fraenkel Set Theory. Although not directly relevant to our concerns, it was soon realized that topological semantics could be unified with Kripke and Beth models, and all generalized, via categorical semantics; see [5] and [10] for good introductions. Here is introduced, not a generalized, but rather an alternative semantics instead. (Incidentally, this semantics can also be understood categorically, as determined by Streicher (unpublished).)

An instance of this semantics was already applied in [9] to the reals as a topological space. The purpose there was to come up with a model of CZF_{Exp} set theory in which the Dedekind cuts do not form a set. CZF_{Exp} contains the Axiom of Exponentiation (the existence of function spaces), but not any stronger Power Set-like axiom, most notably Aczel's Subset Collection, which suffices to

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 309–322, 2009.

prove the Dedekind cuts are a set. The essence of the construction there is that, as in a traditional topological model, the truth value of set membership ($\sigma \in \tau$, where σ and τ are terms) is an open set of $\mathbb{R}$, but at any moment the terms under consideration can collapse to ground model terms. (A ground model term is the canonical image of a ground model set – think of the standard embedding of V into V[G] in classical forcing.) Such a collapse does not make the variable sets disappear, though. So no set could be the Dedekind cuts: any such candidate could at any time collapse to a ground model set, but then it wouldn't contain the canonical generic because that's a variable set, and this generic, over $\mathbb{R}$, is a Dedekind cut.[1]

This process of collapsing to a ground model set we call settling down. Our purpose is to show how this settling semantics works in an arbitrary topological space, not just $\mathbb{R}$. This extension is not completely straightforward. Certain uniformities of $\mathbb{R}$ allowed for simplifications in the definition of forcing ($\Vdash$) and for proofs of stronger set-theoretic axioms, most notably Full Separation and Exponentiation. In the next section, we prove as much as we can making no assumptions on the topological space T being worked over; in the following section, natural and appropriately modest assumptions are made on T so that Separation and Exponentiation can be proven.

The greatest weakness in what can be proven in the general case is in the family of Power Set-like axioms. This is no surprise, as the semantics was developed for a purpose which necessitated the failure of Subset Collection (and hence of Power Set itself). That Exponentiation ended up holding is thanks to the particularities of $\mathbb{R}$, not to settling semantics. Rather, what does hold in general is a weakened version of all of these Power Set-like axioms. The reason that Power Set fails, like the non-existence of the set of Dedekind cuts above, is that any candidate for the power set of X might collapse to a ground model set, and so would then no longer contain any variable subset of X. However, that variable subset might itself collapse, and then would be in the classical power set of X. So while the subset in question, before the collapse, might not equal a member of the classical power set, it cannot be different from every such member. That is the form of Power Set which holds in the settling semantics:

Eventual Power Set: $\forall X\ \exists C\ (\forall Y \in C\ Y \subseteq X) \wedge (\forall Y \subseteq X\ \neg\forall Z \in C\ Y \neq Z)$.

Although we will not need them, there are comparable weakenings of Subset Collection (or Fullness) and Exponentiation:

Eventual Fullness: $\forall X, Y\ \exists C\ (\forall Z \in C\ Z$ is a total relation from X to $Y) \wedge (\forall R$ if R is a total relation from X to Y then $\neg\forall Z \in C\ Z \not\subseteq R)$.

[1] For those already familiar with a similar-sounding construction by Joyal, this is exactly what distinguishes the two. Joyal started with a topological space T, and took the union of T with a second copy of T, the latter carrying the discrete topology (i.e. every subset is open). So by Joyal, you could specialize at a point, but then every set is also specialized there. Here, you can specialize every set you're looking at at a point, but that won't make the ambient variable sets disappear. Alternatively, the whole universe will specialize, but at the same time be reborn. For an exposition of Joyal's argument in print, see either [7] or [17] p. 805-807.

Eventual Exponentiation: $\forall X, Y\ \exists C\ \forall F$ if F is a total function from X to Y then $\neg\forall Z \in C\ F \neq Z$.

It is easy to see that Power Set implies Fullness, which itself implies Exponentiation. Essentially the same arguments will prove:

Proposition 1. *Eventual Power Set implies Eventual Fullness, which in turn implies Eventual Exponentiation.*

As already stated, the original motivation of this work was to generalize an extant construction from one to all topologies. Now that it is done, other uses can be imagined. There has been considerable interest lately in proof-theoretically modest fragments of IZF which are still strong enough to do significant amounts of mathematics, the most prominent of which is CZF (Constructive ZF) [1,2,3,4]. The theory identified here is incomparable with CZF, so its proof-theoretic strength is unclear. If it turns out to be weak, perhaps it could be combined with CZF to provide a slight strengthening of the latter while maintaining a similar proof theory. In any case, the model-theoretic construction might be useful for further independence results, the purpose of the first, motivating model. A long-term project is some kind of classification of models, topological or otherwise; having this unconventional example might help find other yet-to-be-discovered constructions. A question raised by van den Berg is how the model would have to be expanded in order to get a model of IZF. He observed that the recursive realizability model based on (definable subsets of) the natural numbers [8] (also discovered independently by Streicher in unpublished work), which satisfies CZF + Full Separation (and necessarily not Power Set), is essentially just the collection of subcountable sets from the full recursive realizability model [11], and hence is naturally extendable to an IZF model. It is at best unclear how the current model could be so extended. Somewhat speculatively, applications to computer science are also conceivable, wherever such modeling might be natural. For instance, constructive logic can naturally be used to model computation when objects are viewed as having properties only partially determined at any stage; if in addition parallel computation is part of the programming paradigm, it could be that a variable is passed to several parallel sub-computations, which specify the variable more and in incompatible ways. This is similar to the current construction, where there are two transition functions, both leading to the same future but under one function the variable/generic is fully specified and under the other it's not.

2 The General Case

First we define the term structure of the topological model with settling, then truth in the model (the forcing semantics), and then we prove that the model satisfies some standard set-theoretic axioms.

Definition 1. *For a topological space T, a term is a set of the form $\{\langle \sigma_i, J_i\rangle \mid i \in I\} \cup \{\langle \sigma_h, r_h\rangle \mid h \in H\}$, where each σ is (inductively) a term, each J an open set, each r is a member of T, and H and I index sets.*

The first part of each term is as usual. It suffices for the embedding $x \mapsto \hat{x}$ of the ground model into the topological model:

Definition 2. $\hat{x} = \{\langle \hat{y}, T\rangle \mid y \in x\}$. *Any term of the form* $\hat{x}$ *is called a ground model term.*

For ϕ *a formula in the language of set theory with (set, not term) parameters* $x_0, x_1, ..., x_n$, *then* $\hat{\phi}$ *is the formula in the term language obtained from* ϕ *by replacing each* x_i *with* $\hat{x_i}$.

$\check{}$ *is the inverse of* $\hat{}$, *for both sets/terms and formulas:* $\check{\hat{\tau}} = \tau$, $\check{\hat{x}} = x$, $\hat{\check{\phi}} = \phi$, *and* $\check{\hat{\phi}} = \phi$.

The second part of the definition of a term plays a role only when we decide to have the term settle down and stop changing. This settling down in described as follows.

Definition 3. *For a term* σ *and* $r \in T$, σ^r *is defined inductively on the terms as* $\{\langle \sigma_i^r, T\rangle \mid \langle \sigma_i, J_i\rangle \in \sigma \wedge r \in J_i\} \cup \{\langle \sigma_h^r, T\rangle \mid \langle \sigma_h, r\rangle \in \sigma\}$.

Note that σ^r is a ground model term. It bears observation that $(\sigma^r)^s = \sigma^r$.

Definition 4. *For* $\phi = \phi(\sigma_0, ..., \sigma_i)$ *a formula with parameters* $\sigma_0, ..., \sigma_i$, ϕ^r *is* $\phi(\sigma_0^r, ..., \sigma_i^r)$.

We define a forcing relation $J \Vdash \phi$, with J an open subset of T and ϕ a formula.

Definition 5. $J \Vdash \phi$ *is defined inductively on* ϕ:

$J \Vdash \sigma = \tau$ *iff for all* $\langle \sigma_i, J_i\rangle \in \sigma$ $J \cap J_i \Vdash \sigma_i \in \tau$ *and vice versa, and for all* $r \in J$ $\sigma^r = \tau^r$

$J \Vdash \sigma \in \tau$ *iff for all* $r \in J$ *there is a* $\langle \tau_i, J_i\rangle \in \tau$ *and* $J_r \subseteq J_i$ *containing* r *such that* $J_r \Vdash \sigma = \tau_i$

$J \Vdash \phi \wedge \psi$ *iff* $J \Vdash \phi$ *and* $J \Vdash \psi$

$J \Vdash \phi \vee \psi$ *iff for all* $r \in J$ *there is a* $J_r \subseteq J$ *containing* r *such that* $J_r \Vdash \phi$ *or* $J_r \Vdash \psi$

$J \Vdash \phi \rightarrow \psi$ *iff for all* $J' \subseteq J$ *if* $J' \Vdash \phi$ *then* $J' \Vdash \psi$, *and, for all* $r \in J$, *there is a* $J_r \subseteq J$ *containing* r *such that, for all* $K \subseteq J_r$, *if* $K \Vdash \phi^r$ *then* $K \Vdash \psi^r$

$J \Vdash \exists x\ \phi(x)$ *iff for all* $r \in J$ *there is a* $J_r \subseteq J$ *containing* r *and a* σ *such that* $J_r \Vdash \phi(\sigma)$

$J \Vdash \forall x\ \phi(x)$ *iff for all* σ $J \Vdash \phi(\sigma)$, *and for all* $r \in J$ *there is a* $J_r \subseteq J$ *containing* r *such that for all* σ $J_r \Vdash \phi^r(\sigma)$.

(Notice that in the last clause, σ is not interpreted as σ^r.)

Lemma 1. $\Vdash$ *is sound for constructive logic.*

Lemma 2. *T forces the equality axioms, to wit:*

1. $\forall x\ x = x$
2. $\forall x, y\ x = y \rightarrow y = x$
3. $\forall x, y, z\ x = y \wedge y = z \rightarrow x = z$

4. $\forall x, y, z\ x = y \wedge x \in z \rightarrow y \in z$
5. $\forall x, y, z\ x = y \wedge z \in x \rightarrow z \in y$.

Proof. 1: It is trivial to show via a simultaneous induction that, for all J and $\sigma, J \Vdash \sigma = \sigma$, and, for all $\langle \sigma_i, J_i \rangle \in \sigma, J \cap J_i \Vdash \sigma_i \in \sigma$.

2: Trivial because the definition of $J \Vdash \sigma =_M \tau$ is itself symmetric.

3: For this and the subsequent parts, we need a lemma.

Lemma 3. *If $J' \subseteq J \Vdash \sigma = \tau$ then $J' \Vdash \sigma = \tau$, and similarly for $\in$.*

Proof. By induction on σ and τ.

Returning to the main lemma, we show that if $J \Vdash \rho = \sigma$ and $J \Vdash \sigma = \tau$ then $J \Vdash \rho = \tau$, which suffices. This will be done by induction on terms for all opens J simultaneously.

For the second clause in $J \Vdash \rho = \tau$, let $r \in J$. By the hypotheses, second clauses, $\rho^r = \sigma^r$ and $\sigma^r = \tau^r$, so $\rho^r = \tau^r$.

The first clause of the definition of forcing equality follows by induction on terms. Starting with $\langle \rho_i, J_i \rangle \in \rho$, we need to show that $J \cap J_i \Vdash \rho_i \in \tau$. We have $J \cap J_i \Vdash \rho_i \in \sigma$. For a fixed, arbitrary $r \in J \cap J_i$ let $\langle \sigma_j, J_j \rangle \in \sigma$ and $J' \subseteq J \cap J_i$ be such that $r \in J' \cap J_j \Vdash \rho_i = \sigma_j$. By hypothesis, $J \cap J_j \Vdash \sigma_j \in \tau$. So let $\langle \tau_k, J_k \rangle \in \tau$ and $\hat{J} \subseteq J \cap J_j$ be such that $r \in \hat{J} \cap J_k \Vdash \sigma_j = \tau_k$. Let $\tilde{J}$ be $J' \cap \hat{J} \cap J_j$. Note that $\tilde{J} \subseteq J \cap J_i$, and that $r \in \tilde{J} \cap J_k$. We want to show that $\tilde{J} \cap J_k \Vdash \rho_i = \tau_k$. Observing that $\tilde{J} \cap J_k \subseteq J' \cap J_j, \hat{J} \cap J_k$, it follows by the previous lemma that $\tilde{J} \cap J_k \Vdash \rho_i = \sigma_j, \sigma_j =_M \tau_k$, from which the desired conclusion follows by the induction. So $r \in \tilde{J} \cap J_k \Vdash \rho_i \in \tau$. Since $r \in J \cap J_i$ was arbitrary, $J \cap J_i \Vdash \rho_i \in \tau$.

4: It suffices to show that if $J \Vdash \rho = \sigma$ and $J \Vdash \rho \in \tau$ then $J \Vdash \sigma \in \tau$. Let $r \in J$. By hypothesis, let $\langle \tau_i, J_i \rangle \in \tau, J_r \subseteq J_i$ be such that $r \in J_r \Vdash \rho = \tau_i$; without loss of generality $J_r \subseteq J$. By the previous lemma, $J_r \Vdash \rho = \sigma$, and by the previous part of the current lemma, $J_r \Vdash \sigma = \tau_i$. Hence $J_r \Vdash \sigma \in \tau$. Since $r \in J$ was arbitrary, we are done.

5: Similar, and left to the reader.

Lemma 4. 1. *For all ϕ $\emptyset \Vdash \phi$.*
2. *If $J' \subseteq J \Vdash \phi$ then $J' \Vdash \phi$.*
3. *If $J_i \Vdash \phi$ for all i then $\bigcup_i J_i \Vdash \phi$.*
4. *$J \Vdash \phi$ iff for all $r \in J$ there is a $J_r \subseteq J$ containing r such that $J_r \Vdash \phi$.*
5. *For all ϕ, J if $J \Vdash \phi$ then for all $r \in J$ there is a neighborhood J_r of r such that $J_r \Vdash \phi^r$.*
6. *For ϕ bounded (i.e. Δ_0) and having only ground model terms as parameters, $T \Vdash \phi$ iff $\check{\phi}$ (i.e. $V \models \phi$).*

Proof. 1. Trivial induction. This part is not used later, and is mentioned here only to flesh out the picture.

2. Again, a trivial induction. The base cases, $=$ and $\in$, are proven by induction on terms, as mentioned just above.

3. By induction. For the case of $\rightarrow$, you need to invoke the previous part of this lemma. All other cases are straightforward.

4. Trivial, using 3.

5. By induction on ϕ.

$=$: If $r \in J \Vdash \sigma = \tau$ then $\sigma^r = \tau^r$. By the proof of the first part of the equality lemma, $T \Vdash \sigma^r = \tau^r$.

$\in$: If $r \in J \Vdash \sigma \in \tau$, let τ_i, J_i, and J_r be as given by the definition of forcing $\in$. Inductively, some neighborhood of r (or, by the previous case, T itself) forces $\sigma^r = \tau_i^r$. Since $\langle \tau_i^r, T\rangle \in \tau^r$, $T \Vdash \tau_i^r \in \tau^r$, and $T \Vdash \sigma^r \in \tau^r$.

$\vee$: If $r \in J \Vdash \phi \vee \psi$, suppose without loss of generality that $r \in J_r \Vdash \phi$. Inductively let K_r be a neighborhood of r forcing ϕ^r. Then $K_r \Vdash \phi^r \vee \psi^r$.

$\wedge$: If $r \in J \Vdash \phi \wedge \psi$, let J_r and K_r be neighborhoods of r such that $J_r \Vdash \phi$ and $K_r \Vdash \psi$. Then $J_r \cap K_r$ is as desired.

$\rightarrow$: If $r \in J \Vdash \phi \rightarrow \psi$, then J_r as given in the definition of forcing $\rightarrow$ suffices. (To verify the second clause in the definition of $J_r \Vdash \phi^r \rightarrow \psi^r$, use the fact that $(\phi^r)^s = \phi$ and $(\psi^r)^s = \psi$.)

$\exists$: If $r \in J \Vdash \exists x\ \phi(x)$, let $J_r \subseteq J$ and σ be such that $r \in J_r \Vdash \phi(\sigma)$. By induction, let K_r be such that $r \in K_r \Vdash \phi^r(\sigma^r)$. So $K_r \Vdash \exists x\ \phi^r(x)$.

$\forall$: If $r \in J \Vdash \forall x\ \phi(x)$, then J_r as given by the definition of forcing $\forall$ suffices.

6. A simple induction.

At this point, we are ready to show what is in general forced under this semantics.

Theorem 1. *T forces:*

Infinity
Pairing
Union
Extensionality
Set Induction
Eventual Power Set
Bounded (Δ_0) Separation
Collection

Some comments on this choice of axioms are in order. The first five are unremarkable. The role of Eventual Power Set was discussed in the Introduction. The restriction of Separation to the Δ_0 case should be familiar, as that is also the case in CZF and KP. By way of compensation, the version of Collection in CZF is Strong Collection: not only does every total relation with domain a set have a bounding set (regular Collection), but that bounding set can be chosen so that it contains only elements related to something in the domain (the strong version). In the presence of full Separation, these are equivalent, as an appropriate subset of any bounding set can always be taken. Unfortunately, even the additional hypotheses provided by Collection are not enough in the current context to yield even this modest fragment of Separation, as will actually be shown at the beginning of the next section. In fact, even Replacement fails, as we will see.

Proof. – Infinity: $\hat{\omega}$ will do. (Recall that the canonical name $\hat{x}$ of any set x from the ground model is defined inductively as $\{\langle \hat{y}, T\rangle \mid y \in x\}$.)
– Pairing: Given σ and τ, $\{\langle \sigma, T\rangle, \langle \tau, T\rangle\}$ will do.
– Union: Given σ, the union of the following four terms will do:
  • $\{\langle \tau, J \cap J_i\rangle \mid$ for some σ_i, $\langle \tau, J\rangle \in \sigma_i$ and $\langle \sigma_i, J_i\rangle \in \sigma\}$
  • $\{\langle \tau, r\rangle \mid$ for some σ_i, $\langle \tau, r\rangle \in \sigma_i$ and $\langle \sigma_i, r\rangle \in \sigma\}$
  • $\{\langle \tau, r\rangle \mid$ for some σ_i and K, $\langle \tau, K\rangle \in \sigma_i$, $r \in K$, and $\langle \sigma_i, r\rangle \in \sigma\}$
  • $\{\langle \tau, r\rangle \mid$ for some σ_i and K, $\langle \tau, r\rangle \in \sigma_i$, $r \in K$, and $\langle \sigma_i, K\rangle \in \sigma\}$.
– Extensionality: We need to show that $T \Vdash \forall x\, \forall y\, [\forall z\, (z \in x \leftrightarrow z \in y) \to x = y]$. It suffices to show that for any terms σ and τ, $T \Vdash \forall z\, (z \in \sigma \leftrightarrow z \in \tau) \to \sigma = \tau$. (Although that is only the first clause in forcing $\forall$, it subsumes the second, because σ and τ could have been chosen as ground model terms in the first place.) To show that, for the second clause in forcing $\to$, it suffices to show that $T \Vdash \forall z\, (z \in \sigma^r \leftrightarrow z \in \tau^r) \to \sigma^r = \tau^r$. But, as before, this is already subsumed by choosing σ and τ to be ground model terms in the first place. Hence it suffices to check the first clause in forcing $\to$: for all J, if $J \Vdash \forall z\, (z \in \sigma \leftrightarrow z \in \tau)$, then $J \Vdash \sigma = \tau$.
  To this end, let $\langle \sigma_i, J_i\rangle$ be in σ; we need to show that $J \cap J_i \Vdash \sigma_i \in \tau$. By the choice of J, $J \Vdash \sigma_i \in \sigma \leftrightarrow \sigma_i \in \tau$. In particular, $J \Vdash \sigma_i \in \sigma \to \sigma_i \in \tau$. By 4, part 2), $J \cap J_i \Vdash \sigma_i \in \sigma \to \sigma_i \in \tau$. Since $J \cap J_i \Vdash \sigma_i \in \sigma$ (proof of 2, part 1)), $J \cap J_i \Vdash \sigma_i \in \tau$. Symmetrically for $\langle \tau_i, J_i\rangle \in \tau$.
  Also, let $r \in J$. If $\sigma^r \neq \tau^r$, let $\langle \rho, T\rangle$ be in their symmetric difference. By the choice of J, for some neighborhood J_r of r, $J_r \Vdash \rho \in \sigma^r \leftrightarrow \rho \in \tau^r$. This contradicts the choice of ρ. So $\sigma^r = \tau^r$.
– Set Induction (Schema): We need to show that $T \Vdash \forall x\, ((\forall y \in x\ \phi(y)) \to \phi(x)) \to \forall x\ \phi(x)$. The statement in question is an implication. The definition of forcing $\to$ contains two clauses.
  The first clause is that, for any open set J and formula ϕ, if $J \Vdash \forall x(\forall y \in x\ \phi(y) \to \phi(x))$ then $J \Vdash \forall x\ \phi(x)$. By way of proving that, suppose not. Let J and ϕ provide a counter-example. By hypothesis,

$$\forall \sigma\ J \Vdash \forall y \in \sigma\ \phi(y) \to \phi(\sigma) \tag{1}$$

  and

$$\forall r \in J\ \exists J' \ni r\ \forall \sigma'\ J' \Vdash \forall y \in \sigma'\ \phi^r(y) \to \phi^r(\sigma'). \tag{2}$$

  Since J $\not\Vdash \forall x \phi(x)$, either

$$\exists \sigma\ J \not\Vdash \phi(\sigma) \tag{3}$$

  or

$$\exists r \in J\ \forall J' \ni r\ \exists \sigma'\ J' \not\Vdash \phi^r(\sigma'). \tag{4}$$

  If (4) holds, let r as given by (4), and then let J' be as given by (2) for that r. By (4), $\exists \sigma'\ J' \not\Vdash \phi^r(\sigma')$; let σ be such a σ' – so $J' \not\Vdash \phi^r(\sigma)$ – of minimal V-rank. By (2), we have $J' \Vdash \forall y \in \sigma\ \phi^r(y) \to \phi^r(\sigma)$. If we can show that $J' \Vdash \forall y \in \sigma\ \phi^r(y)$, then (by the definition of forcing $\to$) we will have a contradiction, showing that (4) must fail.

To that end, we must show, unpacking the abbreviation, that $J' \Vdash \forall y(y \in \sigma \to \phi^r(y))$; that is,

$$\forall \tau\ J' \Vdash \tau \in \sigma \to \phi^r(\tau) \tag{5}$$

and

$$\forall s \in J'\ \exists K \ni s\ \forall \tau\ K \Vdash \tau \in \sigma^s \to \phi^r(\tau), \tag{6}$$

the latter because $(\phi^r)^s = \phi^r$.

By way of showing (5), suppose $J' \supseteq K \Vdash \tau \in \sigma$. Then K can be covered with open sets K_i such that $K_i \Vdash \tau = \sigma_i$ and $K_i \subseteq J_i$ where $\langle \sigma_i, J_i \rangle \in \sigma$. Since σ_i has strictly lower V-rank than σ, $J' \Vdash \phi^r(\sigma_i)$. Hence $K_i \Vdash \phi^r(\tau)$. Since the K_is cover K (by lemma 4, part 3)) K forces the same. We still have to show that for all $s \in J'$ there is a $K \ni s$ such that for all $K' \subseteq K$ if $K' \Vdash \tau^s \in \sigma^s$ then $K' \Vdash \phi^r(\tau^s)$. In fact, J' suffices for K: if $J' \supseteq K' \Vdash \tau^s \in \sigma^s$ then $K' \Vdash \phi^r(\tau^s)$. Moreover, this is the same argument as the one just completed, with σ^s in place of σ. The only minor observation that bears making is that the V-rank of σ^s is less than or equal to that of σ, so again when τ is forced to be a member of σ^s its V-rank is strictly less than that of σ, so the choice of σ carries us through.

To show (6), we claim that J' suffices for the choice of K: $J' \Vdash \tau \in \sigma^s \to \phi^r(\tau)$. Once more, this is just (5), with σ^s in place of σ.

This completes the proof that (4) must fail. Hence we have that the negation of (4) must hold, namely

$$\forall r \in J\ \exists J' \ni r\ \forall \sigma'\ J' \Vdash \phi^r(\sigma'), \tag{7}$$

as well as (3). Let σ be of minimal V-rank such that $J \not\Vdash \phi(\sigma)$. If we can show that $J \Vdash \forall y \in \sigma\ \phi(y)$, then by (1) we will have a contradiction, completing the proof of the first clause.

What we need to show are

$$\forall \tau\ J \Vdash \tau \in \sigma \to \phi(\tau) \tag{8}$$

and

$$\forall r \in J\ \exists J' \ni r\ \forall \tau\ J' \Vdash \tau \in \sigma^r \to \phi^r(\tau). \tag{9}$$

By way of showing (8), suppose $J \supseteq K \Vdash \tau \in \sigma$; we need to show that $K \Vdash \phi(\tau)$. This is the same argument, based on the minimality of σ, as in the proof of (5). The other part of showing (8) is

$$\forall r \in J\ \exists J' \ni r\ \forall K \subseteq J'\ (K \Vdash \tau^r \in \sigma^r \Rightarrow K \Vdash \phi^r(\tau^r)). \tag{10}$$

Both (9) and (10) are special cases of (7).

This completes the proof of the first clause.

The second clause is that for all $r \in T$ there is a $J \ni r$ such that for all $K \subseteq J$ if $K \Vdash \forall x\ ((\forall y \in x\ \phi^r(y)) \to \phi^r(x))$ then $K \Vdash \forall x\ \phi^r(x)$. For any r, let J be T. Then what remains of the claim has exactly the same form as the first clause, with K and ϕ^r for J and ϕ respectively. Since the validity of this first clause was already shown for all choices of J and ϕ, we are done.

– Eventual Power Set: We need to show that $T \Vdash \forall X\ \exists C\ \forall Y\ (Y \subseteq X \rightarrow \neg\forall Z(Z \in C \rightarrow Y \neq Z))$. (Actually, we must also produce a C that contains only subsets of X. However, to extract such a sub-collection from any C as above is an instance of Bounded Separation, the proof of which below does not rely on the current proof. So we will make our lives a little easier and prove the version of EPS as stated.) Since the sentence forced has no parameters, the second clause in forcing $\forall$ is subsumed by the first, so all we must show is that, for any term σ, $T \Vdash \exists C\ \forall Y\ (Y \subseteq \sigma \rightarrow \neg\forall Z(Z \in C \rightarrow Y \neq Z))$.

  Let $\tau = \{\langle \hat{x}, r\rangle \mid \sigma^r = \hat{s} \wedge x \subseteq s\}$. This is the desired C. It suffices to show that $T \Vdash \forall Y\ (Y \subseteq \sigma \rightarrow \neg\forall Z(Z \in \tau \rightarrow Y \neq Z))$.

  For the first clause in forcing $\forall$, we need to show that $T \Vdash \rho \subseteq \sigma \rightarrow \neg\forall Z(Z \in \tau \rightarrow \rho \neq Z)$. To do that, first suppose $T \supseteq J \Vdash \rho \subseteq \sigma$. (Note that that implies that for all $s \in J$ $T \Vdash \rho^s \subseteq \sigma^s$, so that $\langle \rho^s, s\rangle \in \tau$, and $T \Vdash \rho^s \in \tau^s$.) We must show that $J \Vdash \neg\forall Z(Z \in \tau \rightarrow \rho \neq Z)$. It suffices to show that no non-empty subset K of J forces $\forall Z(Z \in \tau \rightarrow \rho \neq Z)$ or $\forall Z(Z \in \tau^r \rightarrow \rho^r \neq Z)$ $(r \in J)$. For the former, we will show that K must violate the second clause in forcing $\forall$. Let $s \in K$. Letting Z be ρ^s, as just observed, all of T will force $Z \in \tau^s$ but nothing will force $\rho^s \neq Z$. Similarly for the latter, by choosing Z to be ρ^r. To finish forcing the implication, it suffices to show that for all r $T \Vdash \rho^r \subseteq \sigma^r \rightarrow \neg\forall Z(Z \in \tau^r \rightarrow \rho^r \neq Z)$. Again, it suffices to let Z be ρ^r.

  For the second clause in forcing $\forall$, for $r \in T$ and ρ a term, it suffices to show that $T \Vdash \rho \subseteq \sigma^r \rightarrow \neg\forall Z(Z \in \tau^r \rightarrow \rho \neq Z)$. This time letting Z by any ρ^s suffices.

– Bounded Separation: The important point here is that, for ϕ bounded (Δ_0) with only ground model terms, $J \Vdash \phi$ iff $T \Vdash \phi$ iff $V \models \check{\phi}$ (4, part 6).

  We need to show that $T \Vdash \forall X\ \exists Y\ \forall Z\ (Z \in Y \leftrightarrow Z \in X \wedge \phi(Z))$. This means, first, that for any σ, $T \Vdash \exists Y\ \forall Z\ (Z \in Y \leftrightarrow Z \in \sigma \wedge \phi(Z))$, and, second, for any $r \in T$ there is a $J \ni r$ such that, for any σ, $J \Vdash \exists Y\ \forall Z\ (Z \in Y \leftrightarrow Z \in \sigma \wedge \phi^r(Z))$. In the second part, choosing J to be T, we have an instance of the first part, so it suffices to prove the first only.

  Let τ be $\{\langle \sigma_i, J \cap J_i\rangle \mid \langle \sigma_i, J_i\rangle \in \sigma$ and $J \Vdash \phi(\sigma_i)\} \cup \{\langle \hat{x}, r\rangle \mid \langle \hat{x}, T\rangle \in \sigma^r$ and $T \Vdash \phi^r(\hat{x})\}$. We claim that τ suffices: $T \Vdash \forall Z\ (Z \in \tau \leftrightarrow Z \in \sigma \wedge \phi(Z))$.

  First, let ρ be a term. We need to show that $T \Vdash \rho \in \tau \leftrightarrow \rho \in \sigma \wedge \phi(\rho)$. Unraveling the bi-implication and the definition of forcing an implication, that becomes $J \Vdash \rho \in \tau$ iff $J \Vdash \rho \in \sigma \wedge \phi(\rho)$, and $J \Vdash \rho^r \in \tau^r$ iff $J \Vdash \rho^r \in \sigma^r \wedge \phi^r(\rho^r)$. The first iff should be clear from the first part of the definition of τ and the second iff from the second part of the definition, along with the observation that forcing $\phi^r(\rho^r)$ is independent of J.

  We also need, for each $r \in T$, a $J \ni r$ such that for all ρ $J \Vdash \rho \in \tau^r \leftrightarrow \rho \in \sigma^r \wedge \phi^r(\rho)$. Choosing J to be T and unraveling as above (recycling the variable J) yields $J \Vdash \rho \in \tau^r$ iff $J \Vdash \rho \in \sigma^r \wedge \phi^r(\rho)$, and $J \Vdash \rho^s \in \tau^r$ iff $J \Vdash \rho^s \in \sigma^r \wedge \phi^r(\rho^s)$. These hold because the only things that can be forced to be in τ^r or σ^r are (locally) images of ground model terms, and the truth of ϕ^r evaluated at such a term is independent of J.

– Collection: Since only regular, not strong, Collection is true here, it would be easiest to his this with a sledgehammer: reflect V to some set M large enough to contain all the parameters and capture the truth of the assertion in question; the term consisting of the whole universe according to M will be more than enough. It is more informative, though, to follow through the natural construction of a bounding set, so we can highlight in the next section just what goes wrong with the proof of Strong Collection.
We need $T \Vdash \forall x \in \sigma\ \exists y\ \phi(x,y) \rightarrow \exists z\ \forall x \in \sigma\ \exists y \in z\ \phi(x,y)$. It suffices to show that for any J if $J \Vdash \forall x \in \sigma\ \exists y\ \phi(x,y)$ then $J \Vdash \exists z\ \forall x \in \sigma\ \exists y \in z\ \phi(x,y)$, and the same relativized to r. The latter is a special case of the former, so it suffices to show just the former.
By hypothesis, for each $\langle \sigma_i, J_i \rangle \in \sigma$ and $r \in J_i \cap J$ there are τ_{ir} and $J_{ir} \subseteq J_i \cap J$, $J_{ir} \ni r$ such that $J_{ir} \Vdash \phi(\sigma_i, \tau_{ir})$. Also, for all $r \in J$ there is a $J_r \ni r$ such that, for all $\langle \hat{x}, T \rangle \in \sigma^r$, $J_r \Vdash \exists y\ \phi^r(\hat{x}, y)$. For each $s \in J_r$, let $\tau_{r\hat{x}s}$ and $K \ni s$ be such that $K \Vdash \phi^r(\hat{x}, \tau_{r\hat{x}s})$. By 4, part 5), $K \Vdash \phi^r(\hat{x}, \tau^s_{r\hat{x}s})$.
We claim that $\tau = \{\langle \tau_{ir}, J_{ir} \rangle \mid i \in I, r \in J_i \cap J\} \cup \{\langle \tau^s_{r\hat{x}s}, r \rangle \mid r \in J, \langle \hat{x}, T \rangle \in \sigma^r, s \in J_r\}$ suffices: $J \Vdash \forall x \in \sigma\ \exists y \in \tau\ \phi(x,y)$.
Forcing a universal has two parts. The first is that for all ρ, $J \Vdash \rho \in \sigma \rightarrow \exists y \in \tau\ \phi(\rho, y)$. For the second, it suffices to show that for all $r \in J$ and terms ρ $J_r \Vdash \rho \in \sigma^r \rightarrow \exists y \in \tau^r\ \phi^r(\rho, y)$.
For the former, first suppose $J \supseteq K \Vdash \rho \in \sigma$. It should be clear that the first part of τ covers this case. For the other part of forcing that implication, for each $r \in J$, it suffices to show that J_r is as desired: for all $K \subseteq J_r$, if $K \Vdash \rho^r \in \sigma^r$ then $K \Vdash \exists y \in \tau^r\ \phi^r(\rho^r, y)$. This is subsumed by the second implication from above, to which we now turn.
To show $J_r \Vdash \rho \in \sigma^r \rightarrow \exists y \in \tau^r\ \phi^r(\rho, y)$, we need to show first that if $J_r \supseteq K \Vdash \rho \in \sigma^r$ then $K \Vdash \exists y \in \tau^r\ \phi^r(\rho, y)$, and second that for all $s \in J_r$ there is a $K \ni s$ such that if $K \supset L \Vdash \rho^s \in \sigma^r$ then $L \Vdash \exists y \in \tau^r\ \phi^r(\rho^s, y)$. By choosing K to be J_r, the second is subsumed by the first. For that, it should be clear that the second part of τ covers this case. In a bit more detail, it suffices to work locally. (That is, it suffices to find a neighborhood of $s \in K$ forcing what we want, by 4.) Locally, ρ is forced equal to some $\hat{x}$, where $\langle \hat{x}, T \rangle \in \sigma^r$. As already shown, some neighborhood of s forces $\phi^r(\hat{x}, \tau^s_{r\hat{x}s})$, and $\langle \tau^s_{r\hat{x}s}, T \rangle \in \tau^r$ by the second part of τ.

3 Separation and Exponentiation

If Separation were to hold (in the presence of the other axioms from above), then Strong Collection would follow, which itself implies Replacement. Hence a powerful way to show that Separation is not forced is to give an example in which even Replacement fails. In the example below, the offending formula is a Boolean combination of Σ_1 formulas; we do not know if simpler instances of Replacement, such as for Σ_1 or Δ_0 formulas, are falsifiable or instead are actually forced.

Let T_n $(n > 0)$ be the standard space for collapsing $\aleph_n$ to be countable: elements are injections from $\aleph_0$ to $\aleph_n$, an open set is given by a finite partial

function of the same type, an element is in that open set if it is compatible with the partial function. Let T be the disjoint union of the T_ns adjoined with an extra element ∞: $\biguplus_n T_n \cup \{\infty\}$. A basis for the topology is given by all the open subsets of each T_n, plus the basic open neighborhoods of ∞, which are all of the form $\biguplus_{n \geq N} T_n \cup \{\infty\}$ for some N.

This T falsifies Replacement. To state the instance claimed to be falsified, we need several parameters. One is $\{\langle \hat{n}, \infty \rangle \mid n \in \omega\}$, which we will call ω^-. Another is the internalization of the function $n \mapsto \aleph_n$ $(n \in \omega)$, which we will refer to via the free use of the notation $\hat{\aleph}_n$, even when n is just a variable. Finally, we will implicitly need $\hat{\omega}$ in the assertion "X is countable," which is the abbreviation for what you think (i.e. the existence of a bijection with $\hat{\omega}$). Note that "X is uncountable" is taken as the negation of "X is countable."

Proposition 2. *$T \not\Vdash \forall x \in \omega^- \ \exists! y \ [(y = 0 \vee y = 1) \wedge (y = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (y = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)] \rightarrow \exists f \ \forall x \in \omega^- [(f(x) = 0 \vee f(x) = 1) \wedge (f(x) = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (f(x) = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)]$.*

Proof. First we show that T forces the antecedent $\forall x[x \in \omega^- \rightarrow \exists! y \ [(y = 0 \vee y = 1) \wedge (y = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (y = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)]]$.

For the first clause in forcing $\forall$, we need to show that for all σ $T \Vdash \sigma \in \omega^- \rightarrow \exists! y \ [(y = 0 \vee y = 1) \wedge (y = 0 \leftrightarrow \hat{\aleph}_\sigma$ is uncountable$) \wedge (y = 1 \leftrightarrow \neg\neg \hat{\aleph}_\sigma$ is countable$)]$. The first clause in forcing that implication is vacuous, as no open set will force $\sigma \in \omega^-$. The second clause is vacuous for all choices of r except ∞, as then $(\omega^-)^r$ is empty. Finally, for $r = \infty$, it suffices to show that $T \Vdash \exists! y \ [(y = 0 \vee y = 1) \wedge (y = 0 \leftrightarrow \hat{\aleph}_{\hat{n}}$ is uncountable$) \wedge (y = 1 \leftrightarrow \neg\neg \hat{\aleph}_{\hat{n}}$ is countable$)]$. The term which is 0 on $\biguplus_{0<i<n} T_n$ and 1 on the rest of T suffices.

The second clause in forcing $\forall$ is similar.

Since T forces the antecedent of the conditional, it suffices to show that T does not force the consequent: $T \not\Vdash \exists f \ \forall x \in \omega^- [(f(x) = 0 \vee f(x) = 1) \wedge (f(x) = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (f(x) = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)]$. If that were not the case, there would be a term (we will ambiguously refer to as f) and a neighborhood J of ∞ such that $J \Vdash \forall x \in \omega^- [(f(x) = 0 \vee f(x) = 1) \wedge (f(x) = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (f(x) = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)]$. By 4, part 5), there would be a $K \ni \infty$ such that $K \Vdash \forall x \in \hat{\omega} [(f^\infty(x) = 0 \vee f^\infty(x) = 1) \wedge (f^\infty(x) = 0 \leftrightarrow \hat{\aleph}_x$ is uncountable$) \wedge (f^\infty(x) = 1 \leftrightarrow \neg\neg \hat{\aleph}_x$ is countable$)]$. K, being open, contains a set of the form $\biguplus_{n \geq N} T_n$. Let M be $N + 1$. So $K \Vdash (f^\infty(\hat{M}) = 0 \vee f^\infty(\hat{M}) = 1) \wedge (f^\infty(\hat{M}) = 0 \leftrightarrow \hat{\aleph}_{\hat{M}}$ is uncountable$) \wedge (f^\infty(\hat{M}) = 1 \leftrightarrow \neg\neg \hat{\aleph}_{\hat{M}}$ is countable$)$. But $f^\infty(\hat{M})$ is a ground model term, and so is (forced by K to be) equal to $\hat{0}$ or $\hat{1}$. Hence either $K \Vdash \hat{\aleph}_{\hat{M}}$ is uncountable or $K \Vdash \neg\neg \hat{\aleph}_{\hat{M}}$ is countable. But neither is the case, since $K \supseteq T_N \Vdash \hat{\aleph}_{\hat{M}}$ is uncountable and $K \supseteq \biguplus_{n>N} T_n \Vdash \hat{\aleph}_{\hat{M}}$ is countable.

In the example above, the problem around ∞ is that no neighborhood forces just what gets collapsed and what doesn't. It is this lack of homogeneity that is the root cause of the failure of Separation.

Definition 6. *T is locally homogeneous around $r, s \in T$ if there are neighborhoods J_r, J_s of r and s respectively and a homeomorphism of J_r to J_s sending r to s.*

An open set U is homogeneous if it is locally homogeneous around all $r, s \in U$.

T is locally homogeneous if every $r \in T$ has a homogeneous neighborhood.

Lemma 5. *If U is homogeneous, ϕ contains only ground model terms, and $U \supseteq V \Vdash \phi$ (V non-empty), then $U \Vdash \phi$.*

Proof. Let $r \in V$. For $s \in U$, let V_r and V_s be the neighborhoods f the homeomorphism given by the homogeneity of U. $f(\sigma)$ can be defined inductively on terms σ. (Briefly, hereditarily restrict σ to V_r and apply f to the second parts of the pairs in the terms.) $f(\psi)$ is then ψ with f applied to the parameters. It is easy to show inductively on formulas that $V_r \Vdash \psi$ iff $V_s \Vdash f(\psi)$.

If ϕ contains only ground model terms, then $f(\phi) = \phi$. So U is covered by open sets that force ϕ. Hence $U \Vdash \phi$.

Theorem 2. *If T is locally homogeneous then $T \Vdash FullSeparation$.*

Proof. As in the proof of Bounded Separation from the previous section, we have to show that, for any σ, $T \Vdash \exists Y\ \forall Z\ (Z \in Y \leftrightarrow Z \in \sigma \wedge \phi(Z))$, only this time with no restriction on ϕ. The choice of witness Y is slightly different. For each r let $K_r \ni r$ be homogeneous. Let τ be $\{\langle \sigma_i, J \cap J_i\rangle \mid \langle \sigma_i, J_i\rangle \in \sigma$ and $J \Vdash \phi(\sigma_i)\} \cup \{\langle \hat{x}, r\rangle \mid \langle \hat{x}, T\rangle \in \sigma^r$ and $K_r \Vdash \phi^r(\hat{x})\}$. The difference from before is that in the latter part of τ membership is determined by what's forced by K_r instead of by T. We claim that τ suffices: $T \Vdash \forall Z\ (Z \in \tau \leftrightarrow Z \in \sigma \wedge \phi(Z))$.

For the first clause in forcing $\forall$, let ρ be a term. We need to show $T \Vdash \rho \in \tau \leftrightarrow \rho \in \sigma \wedge \phi(\rho)$. By the first clause in forcing $\rightarrow$, we have to show that for all J $J \Vdash \rho \in \tau$ iff $J \Vdash \rho \in \sigma \wedge \phi(\rho)$, which should be clear from the first part of τ. For the second clause in $\rightarrow$ it suffices to show that for all $J \subseteq K_r$ $J \Vdash \rho^r \in \tau^r$ iff $J \Vdash \rho^r \in \sigma^r \wedge \phi^r(\rho^r)$. Regarding forcing membership, all of the terms here are ground model terms, so membership is absolute (does not depends on the choice of J). If ρ^r enters τ^r because of the first part of τ's definition, then we have $\sigma_i^r = \rho^r$, $r \in J \Vdash \phi(\sigma_i)$, $r \in J_i$, and $\langle \sigma_i, J_i\rangle \in \sigma$. By 4, part 5), some neighborhood J_r of r forces $\phi^r(\sigma_i^r)$. By the lemma just above (applied to $K_r \cap J_r$), K_r forces the same. Hence we can restrict our attention to terms ρ^r which enter τ because of τ's definition's second part. Again by the preceding lemma, for J non-empty, $J \Vdash \phi^r(\rho^r)$ iff $K_r \Vdash \phi^r(\rho^r)$, which suffices. (For J empty, J forces everything.)

For the second clause in forcing $\forall$, it suffices to show that $K_r \Vdash \rho \in \tau^r \leftrightarrow \rho \in \sigma^r \wedge \phi^r(\rho)$. If any $J \subseteq K_r$ forces $\rho \in \tau^r$ or $\rho \in \sigma^r$, then locally ρ is forced to be some ground model term, and we're in the same situation as in the previous paragraph.

It would be nice to turn the previous theorem into an iff. If that is false, it would be interesting to see exactly what condition is equivalent to Full Separation. Presumably it would have something to do with homogeneity, since the proof

given seems so natural, but it's possible that the correct condition, if weaker than local homogeneity, would involve different issues. It's also possible that Separation has no natural correspondent on the topological side, which would be very unfortunate, but still important to know.

We now turn our attention to Exponentiation.

Theorem 3. *If T is locally connected, then $T \Vdash$ Exponentiation.*

Proof. Given terms σ and χ, let τ be $\{\langle \rho, J\rangle \mid J \Vdash \rho$ is a function from σ to $\chi\} \cup \{\langle \hat{x}, r\rangle \mid x$ is a function from $\check{\sigma}^r$ to $\check{\chi}^r\}$. (τ can be arranged to be set-sized by requiring that ρ be hereditarily empty outside of J.) It suffices to show that $T \Vdash \forall z\ (z \in \tau \leftrightarrow z$ is a function from σ to $\chi)$.

The first clause in forcing $\forall$ is that, for any term ρ, $T \Vdash \rho \in \tau \leftrightarrow \rho$ is a function from σ to χ. That $J \Vdash \rho \in \tau$ iff $J \Vdash$ "ρ is a function from σ to χ" is immediate from the first part of τ. As for $J \Vdash \rho^r \in \tau^r$ iff $J \Vdash$ "ρ^r is a function from σ^r to χ^r", by 4, part 6), both of those statements are independent of J, and the iff holds because of the second part of τ.

The crux of the matter is the second clause in forcing $\forall$: $J \Vdash \rho \in \tau^r$ iff $J \Vdash$ "ρ is a function from σ^r to χ^r". Why can only ground model functions be forced (locally) to be functions? For $s \in J$, let $K_s \subseteq J$ be a connected neighborhood of s. For each $\langle \sigma_i, T\rangle \in \sigma^r$, pick a $\langle \chi_i, T\rangle \in \chi$ such that the value of (i.e. the largest subset of K_s forcing) "$\rho(\sigma_i) = \chi_i$" is non-empty. That set, along with the value of "$\rho(\sigma_i) \neq \chi_i$", is a disjoint open cover of K_s. Since K_s is connected, the latter set is empty. So all of the values of ρ are determined by K_s, so K_s forces ρ to equal a ground model term. Since J is covered by such sets, J also forces ρ to be a ground model term.

Again, it would be nice to turn this into an iff, or, failing that, to know what topological equivalent there is to Exponentiation.

An application of these theorems can be found in [9]. The second model presented there is the topological semantics of the current paper applied to $\mathbb{R}$ (with the standard topology). $\mathbb{R}$ is homogeneous (not just locally so) and locally connected, which is why that model satisfied Separation and Exponentiation. An example where Exponentiation fails is if T is Cantor space. Forcing with T produces a random 0-1 sequence, which is a function from $\mathbb{N}$ to 2. So the canonical generic is in a function space, but cannot be captured by any ground model set.

References

1. Aczel, P.: The type theoretic interpretation of constructive set theory. In: MacIntyre, A., Pacholski, L., Paris, J. (eds.) Logic Colloquium 1977, pp. 55–66. North-Holland, Amsterdam (1978)
2. Aczel, P.: The type theoretic interpretation of constructive set theory: choice principles. In: Troelstra, A.S., van Dalen, D. (eds.) The L.E.J. Brouwer Centenary Symposium, pp. 1–40. North-Holland, Amsterdam (1982)
3. Aczel, P.: The type theoretic interpretation of constructive set theory: inductive definitions. In: Marcus, R.B., et al. (eds.) Logic, Methodology and Philosophy of Science VII, pp. 17–49. North-Holland, Amsterdam (1986)

4. Aczel, P., Rathjen, M.: Notes on constructive set theory, Technical Report 40, 2000/2001. Mittag-Leffler Institute, Sweden (2001)
5. Fourman, M.P., Scott, D.S.: Sheaves and Logic. In: Fourman, M.P., Mulvey, C.J., Scott, D.S. (eds.) Applications of Sheaves. Lecture Notes in Mathematics, vol. 753, pp. 302–401. Springer, Heidelberg (1979)
6. Grayson, R.J.: Heyting-valued models for intuitionistic set theory. In: Fourman, M.P., Mulvey, C.J., Scott, D.S. (eds.) Applications of Sheaves. Lecture Notes in Mathematics, vol. 753, pp. 402–414. Springer, Heidelberg (1979)
7. Grayson, R.J.: Heyting-valued semantics. In: Lolli, G., Longo, G., Marcja, A. (eds.) Logic Colloquium 1982. Studies in Logic and the Foundations of Mathematics, vol. 112, pp. 181–208. North-Holland, Amsterdam (1984)
8. Lubarsky, R.: CZF and Second Order Arithmetic. Annals of Pure and Applied Logic 141, 29–34 (2006)
9. Lubarsky, R., Rathjen, M.: On the constructive Dedekind reals. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 349–362. Springer, Heidelberg (2007); Also Logic and Analysis, vol. 1, pp. 131–152 (2008)
10. MacLane, S., Moerdijk, I.: Sheaves in Geometry and Logic. Springer, New York (1992)
11. McCarty, D.: Realizability and recursive mathematics. Ph.D. thesis, Oxford University (1984), also Carnegie-Mellon University Technical Report CMU-CS-84-131
12. Mostowski, A.W.: Proofs of non-deducibility in intuitionistic functional calculus. JSL 13, 204–207 (1948)
13. Scott, D.S.: Extending the topological interpretation to intuitionistic analysis I. Compos. Math. 20, 194–210 (1968)
14. Scott, D.S.: Extending the topological interpretation to intuitionistic analysis II. In: Myhill, J., Kino, A., Vesley, R.E. (eds.) Intuitionism and Proof Theory, pp. 235–255. North-Holland, Amsterdam (1970)
15. Stone, M.H.: Topological representations of distributive lattices and Brouwerian logics. Casopis Pro Pestvovani Mathematiky a Fysiki Cast Matematicka 67, 1–25 (1937)
16. Tarski, A.: Der Aussagenkalkül and die Topologie. Fundam. Math. 31, 103–134 (1938)
17. Troelstra, A.S., van Dalen, D.: Constructivism in Mathematics – An Introduction, vol. II. In: Studies in Logic and the Foundations of Mathematics, vol. 123. North-Holland, Amsterdam (1988)

Automata and Answer Set Programming

Victor Marek[1] and Jeffrey B. Remmel[2]

[1] Department of Computer Science, University of Kentucky 40506
marek@cs.uky.edu.edu
[2] Departments of Mathematics and Computer Science, University of California at San Diego, La Jolla, CA 92903
jremmel@ucsd.edu

Abstract. In answer set programming (ASP), one does not allow the use of function symbols. Disallowing function symbols avoids the problem of having logic programs which have stable models of excessively high complexity. For example, Marek, Nerode, and Remmel showed that there exist finite predicate logic programs which have stable models but which have no hyperarithmetic stable model. Of course, by eliminating function symbols, one loses a lot of expressive power in the language. In particular, it is difficult to directly reason about infinite sets in ASP.

Blair, Marek, and Remmel [BMR08] developed an extension of logic programming called *set based logic programming*. In the theory of set based logic programming, the atoms represent subsets of a fixed universe X and one is allowed to compose the one-step consequence operator with a monotonic idempotent operator O so as to ensure that the analogue of stable models are always closed under O. We show that if the sets represented by the atoms in a finite set based program P are languages accepted by finite automaton, and the operators involved in the construction have a certain natural property, then all the stable models of P are languages accepted by finite automaton and one can effectively check whether a language accepted by a finite automaton is a stable model of the set based logic program. Thus in this setting, one can effectively reason about certain classes of infinite sets.

1 Introduction

Computer Science for the most part reasons about finite sets, relations and functions. There are many examples in computer science where adding symbols for infinite sets or arbitrary function symbols into programming languages results in big jumps in the complexity of models of programs. For example, finding the least model of a finite Horn program with no function symbols can be done in linear time [DG82] while the least model of finite predicate logic Horn program with function symbols can be an arbitrary recursively enumerable set [Sm68]. If we consider logic programs with negation, Marek and Truszczyński [MT93] showed that the question of whether a finite propositional logic program has a stable model is NP-complete. However Marek, Nerode, and Remmel [MNR94] showed that the question of whether a finite predicate logic program with function symbols possesses a stable model is Σ^1_1 complete. Similarly, the stable models of

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 323–337, 2009.

logic programs that contain function symbols can be quite complex. Starting with [AB90] and continuing with [BMS95] and [MNR94], a number of results showed that the stable models of logic programs that allow function symbols can be exceedingly complex, even in the case where the program has a unique stable model. For example, Marek, Nerode and Remmel [MNR94] showed that there exist finite predicate logic programs which have stable models but which have no hyperarithmetic stable model.

While these type of results may at first glance appear negative, they had a positive effect in the long run since they forced researchers and designers to limit themselves to cases where programs can be actually processed. The effect was that processing programs called *solvers* such as cmodels [BL02, GLM06], smodels [SNS02], *clasp* [GKN+], ASSAT [LZ02], and `dlv` [LPF+] had to focus on finite programs that do not admit function symbols. The designers of solvers have also focused on the issues of both improving processing of the logic programs (i.e. searching for a stable model) and improving the use of logic programs as a programming language. The latter task consists of extending the constructs available to the programmer to make programming easier and more readable. This resulted in a class of solvers that found use in combinatorial optimization, hardware verification and other applications.

Of course, by eliminating function symbols, one loses a lot of expressive power in the language. One of the motivations of this paper was to find ways to extend the ASP formalism to allow one to reason *directly* about infinite sets yet still allow the programs to be processed in an effective manner. This requires a very careful analysis of the complexity issues involved in the formalisms as well as developing various ways to code the infinite sets involved in any given application so that one can process information effectively. Part of our motivation is that with the rise of the Internet, there are now many tools which use the Internet as a virtual data base. While all the information on the Internet at any given point of time is a finite object, it is constantly changing and it would be nearly impossible to characterize the totality of information available in any meaningful way. Thus, for all practical purposes, one can consider the information on the Internet as an infinite set of information. Hence we need to consider ways in which one can extend various formalisms in computer science to reason about infinite objects.

The main goal of this paper is to show that there are extensions of the ASP formalism where one can effectively reason about infinite languages which are accepted by deterministic finite automata (DFAs). In particular, we shall show that in a recent extension of logic programming due to Blair, Marek, and Remmel [BMR08], one can effectively reason about languages which are accepted by finite automaton. That is, in [BMR01], Blair, Marek, and Remmel developed an extension of the logic programming paradigm called *spatial logic programming* in which one can directly reason about regions in space and time as might be required in applications like graphics, image compression, or job scheduling. In spaital logic programming, one has some fixed space X be the intended universe of the program rather than having the Herbrand base be the intended underlying

universe of the program and one has each atom of the language of the program specify a subset of X, i.e. an element of the set 2^X.

As pointed out in [BMR08], if one reflects for a moment on the basic aspects of logic programming with an Herbrand model interpretation, a slight change in one's point of view shows that it is natural to interpret atoms as subsets of the Herbrand base. In ordinary logic programming, we determine the truth value of an atom p in an Herbrand interpretation I by declaring $I \models p$ if and only if $p \in I$. However, this is equivalent to defining the *sense*, $[\![p]\!]$, of an atom p to be the set $\{p\}$ and declaring that $I \models p$ if and only if $[\![p]\!] \subseteq I$. By this simple move, we have permitted ourselves to interpret the sense of an atom as a subset of a set X rather than the literal atom itself in the case where X is the Herbrand base of the language of the program.

It turns out that if the underlying space X has structure such as a topology or an algebraic structure such as a group or vector space, then a number of other natural options present themselves. That is, Blair, Marek, and Remmel [BMR08] extended the theory of spatial logic programming to what they called *set based logic programming* where one composes the one-step consequence operator of spatial logic programing with a monotonic idempotent operator. For example, if we are dealing with a topological space, one can construct a new one-step consequence operator T by composing the one-step consequence operator for spatial logic programming with an operator that produces the topological closure of a set or the interior of a set. In such a situation, we can ensure that the new one-step consequence operator T *always* produces a closed set or always produces an open set. We say that an operator $O : 2^X \to 2^X$ is *monotonic* if for all $Y \subseteq Z \subseteq X$, we have $O(Y) \subseteq O(Z)$ and we say that O is *idempotent* for all $Y \subseteq X$, $O(O(Y)) = O(Y)$. Specifically, many familiar operators such as *closure*, *interior*, or the *span* and *convex-closure* operators in vector spaces over the rationals are monotonic idempotent operators. We call a monotonic idempotent operator a *miop*. We say that a set Y is *closed* with respect to miop O if and only if $Y = O(Y)$. By composing the one-step consequence operator for spatial logic programs with the operator O, we can ensure that the resulting one-step consequence operator always produces a fixed point of O.

Moreover, in such a setting, one also has a variety of options for how to interpret negation. In normal logic programming, a model M satisfies $\neg p$ if $p \notin M$. From the spatial logic programming point of view, when p is interpreted as a singleton $\{p\}$, this would be equivalent to saying that M satisfies $\neg p$ if (i) $\{p\} \cap M = \emptyset$, or (equivalently) (ii) $\{p\} \not\subseteq M$. When the sense of p is a set with more than one element, it is easy to see that saying that M satisfies $\neg p$ if $[\![p]\!] \cap M = \emptyset$ (strong negation) is different from saying that M satisfies $\neg p$ if $[\![p]\!] \not\subseteq M$ (weak negation). This leads to two natural interpretations of the negation symbol which are compatible with the basic logic programming paradigm. When the underlying space has a miop cl, one can get even more subsidiary types of negation by taking M to satisfy $\neg p$ if $cl([\![p]\!]) \cap M \subseteq cl(\emptyset)$ (strong negation) or by taking M to satisfy $\neg p$ if $cl([\![p]\!]) \not\subseteq M$ (weak negation).

Blair, Marek, and Remmel [BMR08] showed that set based logic programing provides the foundations and basic techniques for crafting applications in the *answer set paradigm* as described in [MT99, Nie99] and then [GL02, Ba03]. The expressive power of miops allows us to capture functions and relations intrinsic to the domain of a spatial logic program, but independent of the program. This permits set based logic programs to seamlessly serve as front-ends to other systems. Miops play the role of back-end, or "behind-the-scenes", procedures and functions.

The main goal of this paper is to show that the theory of deterministic finite automata (DFA's) can be integrated with the theory of set based logic programming to give a setting where one can effectively reason about infinite sets. That is, suppose that P is a finite set based logic program over a universe X where the sets represented by atoms in P are languages contained in X which are accepted by finite automaton and the miops O involved in P preserve regular languages, i.e, if A is an automata such that the language $L(A)$ accepted by A is contained in X, then we can effectively construct an automaton B such that the language $L(B)$ accepted by B equals $O(L(A))$.Then, we shall show that the stable models of P are languages accepted by finite automaton and one can effectively check whether a language accepted by finite automaton is a stable model. Thus in this setting, one can effectively reason about infinite sets.

The outline of this paper is as follows. In section 2, we shall give the basic definitions of set based logic programming with miops. In section 3, we shall review that basic properties of languages accepted by finite automata. In section 4, we shall show how the formalisms of finite automata can be incorporated into the set based logic programming. Finally, in section 5, we give conclusions and directions for further research.

2 Set Logic Programs: Syntax, Miops, and Semantics

We review the basic definitions of set based logic programming as introduced by Blair, Marek, and Remmel [BMR08]. The syntax of set based logic programs will essentially be the syntax of DATALOG programs with negation.

A **set based augmented first-order language** (**set based language**, for short) $\mathcal{L}$ is a triple $(L, X, [\![\cdot]\!])$, where

(1) L is a language for first-order predicate logic (without function symbols other than constants),
(2) X is a nonempty (possibly infinite) set, called the **interpretation space**, and
(3) $[\![\cdot]\!]$ is a mapping from the atoms of L to the power set of X, called the *sense assignment*. If p is an atom, then $[\![p]\!]$ is called the *sense* of p.

A **set based logic program** has three components.

1) The language $\mathcal{L}$ which includes the interpretation space and the sense assignment.
2) The IDB (**Intentional Database**): A finite set of program clauses, each of the form $A \leftarrow L_1, \ldots, L_n$, where each L_i is a *literal*, i.e. an atom or the negation of an atom, and A is an atom.

3) The EDB (**Extensional Database**): A finite set of clauses of the form $A \leftarrow$ where A is an atom.

Given a set based logic program P, the *Herbrand base* of P is the Herbrand base of the smallest set based language over which P is a set based logic program.

We shall assume that the classes of set based logic programs that we consider are always over a language for first-order logic L with no function symbols except constants, and a fixed set X. We let HB_L denote the Herbrand base of L, i.e. the set of atoms of L. We omit the subscript L when the context is clear. Thus we allow clauses whose instances are of the following form:

$$\mathcal{C} = A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m. \tag{1}$$

where A, B_i, and C_j are atoms for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. We let $head(\mathcal{C}) = A$, $Body(\mathcal{C}) = B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m$, and $PosBody(\mathcal{C})$=$\{B_1, \ldots, B_m\}$, and $NegBody(\mathcal{C}) = \{C_1, \ldots, C_m\}$.

We let 2^X be the powerset of X. Given $[\![\cdot]\!] : \mathrm{HB}_L \longrightarrow 2^X$, an *interpretation* I of the set based language $\mathcal{L} = (L, X, [\![\cdot]\!])$ is a subset of X.

2.1 Examples of Monotonic Idempotent Operators

A second component of a set based logic program is one or more monotonic idempotent operators $O : 2^X \to 2^X$ that are associated with the program.

For example, suppose that the interpretation space X is either $\mathbf{R}^n$ or $\mathbf{Q}^n$ where $\mathbf{R}$ is the reals and $\mathbf{Q}$ is the rationals. Then, X is a topological vector space under the usual topology so that we have a number of natural miop operators:

1. $op_{id}(A) = A$, i.e. the identity map is simplest miop operator,
2. $op_{c}(A) = \bar{A}$ where $\bar{A}$ is the smallest closed set containing A,
3. $op_{int}(A) = int(A)$ where $int(A)$ is the interior of A,
4. $op_{convex}(A) = K(A)$ where $K(A)$ is the convex closure of A, i.e. the smallest set $K \subseteq X$ such that $A \subseteq K$ and whenever $x_1, \ldots, x_n \in K$ and $\alpha_1, \ldots, \alpha_n$ are elements of the underlying field ($\mathbf{R}$ or $\mathbf{Q}$) such that $\sum_{i=1}^{n} \alpha_i = 1$, then $\sum_{i=1}^{n} \alpha_i x_i$ is in K, and
5. $op_{subsp}(A) = (A)^*$ where $(A)^*$ is the subspace of X generated by A.

We should note that (5) is a prototypical example if we start with an *algebraic* structure. That is, in such cases, we can let $op_{substr}(A) = (A)^*$ where $(A)^*$ is the substructure of X generated by A. Examples of such miops include the following:

(a) if X is a group, we can let $op_{subgrp}(A) = (A)^*$ where $(A)^*$ is the subgroup of X generated by A,
(b) if X is a ring, we can let $op_{subrg}(A) = (A)^*$ where $(A)^*$ is the subring of X generated by A,
(c) if X is a field, we can let $op_{subfld}(A) = (A)^*$ where $(A)^*$ is the subfield of X generated by A,
(d) if X is a Boolean algebra, we can let $op_{subalg}(A) = (A)^*$ where $(A)^*$ is the subalgebra of X generated by A or we can let $op_{ideal}(A) = Id(A)$ where $Id(A)$ is the ideal of X generated by A, and

(e) if $(X, \leq_X)$ is a partially ordered set, we can let $op_{uideal}(A) = Uid(A)$ where $Uid(A)$ is the upper order ideal of X (that is, the least subset S of X containing A such that whenever $x \in S$ and $x \leq_X y$, then $y \in S$).

2.2 Set Based Logic Programming with Miops

Now suppose that we are given a miop $op^+ : 2^X \to 2^X$ and Horn set based logic program P over X. Here we say that a set based logic program is *Horn* if its IDB is Horn. Blair, Marek, and Remmel [BMR08] generalized the one-step consequence-operator of ordinary logic programs with respect to 2-valued logic to set based logic programs relative to a miop operator op^+ as follows. First, for any atom A and $I \subseteq X$, we say that $I \models_{[\![\cdot]\!],op^+} A$ if and only if $op^+([\![A]\!]) \subseteq I$. Then, given a set based logic program P with IDB P, let P' be the set of instances of a clauses in P and let

$$T_{P,op^+}(I) = op^+(I_1 \cup I_2)$$

where $I_1 = \bigcup\{[\![a]\!] \mid a \leftarrow L_1, \ldots, L_n \ \in P', I \models_{[\![\cdot]\!],op^+} L_i, i = 1, \ldots, n\}$ and
$I_2 = \bigcup\{[\![a]\!] \mid a \leftarrow$ is an instance of a clause in the EDB of $P\}$.
We then say that a *supported model relative to* op^+ of P is a fixed point of T_{P,op^+}.

We iterate T_{P,op^+} according to the following.

$$\begin{aligned} T_{P,op^+} \uparrow^0 (I) &= I \\ T_{P,op^+} \uparrow^{\alpha+1} (I) &= T_{P,op^+}(T_{P,op^+} \uparrow^\alpha (I)) \\ T_{P,op^+} \uparrow^\lambda (I) &= op^+(\bigcup_{\alpha<\lambda} \{T_{P,op^+} \uparrow^\alpha (I)\}),\ \lambda \text{ limit} \end{aligned}$$

It is easy to see that if P is a Horn spatial logic program and op^+ is a miop, then T_{P,op^+} is monotonic. Blair, Marek, and Remmel [BMR08] proved the following.

Theorem 1. Given a miop op^+, the least model of a Horn set based logic program P exists and is closed under op^+ , is supported relative op^+, and is given by $\mathbf{T}_{P,op^+} \uparrow^\alpha (\emptyset)$ for the least ordinal α at which a fixed point is obtained.

We note, however, that if the Herbrand universe of a set based logic program is infinite (contains infinitely many constants) then, unlike the situation with ordinary Horn programs, T_{P,op^+} will not in general be upward continuous even in the case where $op^+(A) = A$ for all $A \subseteq X$. That is, consider the following example which was given in [BMR08].

Example 1. Assume that op^+ is the identity operator on 2^X. To specify a set based logic program, we must specify the language, EDB and IDB. Let $\mathcal{L} = (L, X, [\![\cdot]\!])$ where L has four unary predicate symbols: p, q, r and s, and countably many constants $e_0, e_1, \ldots,$. X is the set $\mathbf{N} \bigcup \{\mathbf{N}\}$ where $\mathbf{N}$ is the set of natural numbers, $\{0, 1, 2, \ldots\}$. $[\![\cdot]\!]$ is specified by $[\![q(e_n)]\!] = \{0, \ldots, n\}$, $[\![p(e_n)]\!] = \{0, \ldots, n+1\}$, $[\![r(e_n)]\!] = \mathbf{N}$, and $[\![s(e_n)]\!] = \{\mathbf{N}\}$.

The EDB is $q(e_0) \leftarrow$ and the IDB is: $p(X) \leftarrow q(X)$ and $s(e_0) \leftarrow r(e_0)$.

Now, after ω iterations upward from the empty interpretation, $r(e_0)$ becomes satisfied. One more iteration is required to reach an interpretation that satisfies $s(e_0)$, where the least fixed point is attained. □

Next we consider how we should deal with negation in the setting of miop operators. Suppose that we have a miop operator op^- on the space X. We do not require that op^- is the same as that miop op^+ but it may be. Our goal is to define two different satisfaction relations for negative literals relative to the miop operator op^- which are called strong and weak negation in [BMR08] [1].

For the rest of this paper, we shall think of a set based logic program P as a set of clauses of the form (1) where it may be that either n or m equals 0. We let $horn(P)$ denote the set of all Horn clauses in P and $nohorn(P) = P - horn(P)$.

Definition 1. Suppose that P is a set based logic program over X and op^+ and op^- are miops on X and $a \in \{s, w\}$.

(I) Given any atom A and set $J \subseteq X$, then we say
$J \models^a_{[\![\cdot]\!], op^+, op^-} A$ if and only if $op^+([\![A]\!]) \subseteq J$.

(II)$_s$ (Strong negation) Given any atom A and set $J \subseteq X$, then we say
$J \models^s_{[\![\cdot]\!], op^+, op^-} \neg A$ if and only if $op^-([\![A]\!]) \cap J \subseteq op^-(\emptyset)$.

(II)$_w$ (Weak negation) Given any atom A and set $J \subseteq X$, then we say
$J \models^w_{[\![\cdot]\!], op^+, op^-} \neg A$ if and only if $op^-([\![A]\!]) \not\subseteq J$.

Definition 2. For any given set $J \subseteq X$ we define the *strong Gelfond-Lifschitz transform*, $GL^s_{J,[\![\cdot]\!], op^+, op^-}(P)$, of a program P with respect to miops op^+ and op^- on 2^X, in two steps. First, we consider all clauses in P,

$$\mathcal{C} = A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, C_m \tag{2}$$

where $A, B_1, \ldots, B_n, C_1, \ldots, C_m$ are atoms. If for some i, it is *not* the case that $J \models^s_{[\![\cdot]\!], op^+, op^-} \neg C_i$, then we eliminate clause $\mathcal{C}$. Otherwise we replace $\mathcal{C}$ by the Horn clause

$$A \leftarrow B_1, \ldots, B_n. \tag{3}$$

Then, $GL^s_{J,[\![\cdot]\!], op^+, \mathcal{R}}(P)$ consists of the set of all Horn clauses produced by this two step process.

We define the *weak Gelfond-Lifschitz transform*, $GL^w_{J,[\![\cdot]\!], op^+, op^-}(P)$, of a program P with respect to miops op^+ and op^- on 2^X in a similar manner except that we use $\models^w_{[\![\cdot]\!], op^+, op^-}$ in place of $\models^s_{[\![\cdot]\!], op^+, op^-}$ in the definition.

[1] Lifschitz [Li94] observed that different modalities, thus different operators, can be used to evaluate positive and negative part of bodies of clauses of normal programs.

Note that since $GL^a_{J,[\![\cdot]\!],op^+,op^-}(P)$ is a Horn set based logic program for either $a = s$ or $a = w$, the least model of $GL^a_{J,[\![\cdot]\!],op^+,op^-}(P)$ relative to op^+ is defined. We then define the a-stable model semantics for a set based logic program P over X relative to the miops op^+ and op^- on X for $a \in \{s, w\}$ as follows.

Definition 3. J is an *a-stable* model of P *relative* to op^+ and op^- if and only if J is the least fixed point of $T_{GL^a_{J,[\![\cdot]\!],op^+,op^-}(P),op^+}$.

Next we give a simple example to show that there is a difference between s-stable and w-stable models.

Example 2. Suppose that the space $X = \mathcal{R}^2$ is the real plane. Our program will have two atoms $\{a, b\}, \{c, d\}$ where a, b, c and d are reals. We let $[a, b]$ and $[c, d]$ denote the line segments connecting a to b and c to d respectively. We let the sense of the these atoms be the corresponding subsets, i.e. we let $[\![\{a, b\}]\!] = \{a, b\}$ and $[\![\{c, d\}]\!] = \{c, d\}$. We let $op^+ = op^- = op_{convex}$. The consider the following program $\mathcal{P}$.

(1) $\{a, b\} \leftarrow \neg\{c, d\}$
(2) $\{c, d\} \leftarrow \neg\{a, b\}$

There are four possible candidate for stable models in this case, namely (i) $\emptyset$, (ii) $[a, b]$, (iii) $[c, d]$, and (iv) $op_{convex}\{a, b, c, d\}$. Let us recall that $op_{convex}(X)$ is the convex closure of X which, depending on a, b, c, and d may be either a quadrilateral, triangle, or a line segment.

If we are considering s-stable models where $J \models^s_{[\![\cdot]\!],op^+,op^-} \neg C$ if and only if $op^-(C) \cap J = op^-(\emptyset) = \emptyset$, then the only case where there are s-stable models if $[a, b]$ and $[c, d]$ are disjoint in which (ii) case and (iii) are s-stable models.

If we are considering w-stable models where $J \models^w_{[\![\cdot]\!],op^+,op^-} \neg C$ if and only if $op^-(C) \not\subseteq J$, then there are no w-stable models if $[a, b] = [c, d]$, (ii) is a w-stable model if $[a, b] \not\subseteq [c, d]$, (iii) is w-stable model if $[c, d] \not\subseteq [a, b]$ and (ii) and (iii) are w-stable models if neither $[a, b] \subseteq [c, d]$ nor $[c, d] \subseteq [a, b]$. △

It is still the case that the a-stable models of a set based logic program P form an antichain for $a \in \{s, w\}$. That is, we have the following result.

Theorem 2. *Suppose that P is a set based logic program over X, op^+ and op^- are miops on X, and $a \in \{s, w\}$. If M and N are a-stable models of P and $M \subseteq N$, then $M = N$.*

Proof. It is easy to see that in general if $M \subseteq N$, then

$$GL^a_{N,[\![\cdot]\!],op^+,op^-}(P) \subseteq GL^a_{M,[\![\cdot]\!],op^+,op^-}(P).$$

Hence the least fixed point of $T_{GL^a_{N,[\![\cdot]\!],op^+,op^-}(P),op^+}$ is a subset of the least fixed point of $T_{GL^x_{M,[\![\cdot]\!],op^+,op^-}(P),op^+}$. But if $M \subseteq N$ and M and N are a-stable models, then N equals the least fixed point of $T_{GL^a_{N,[\![\cdot]\!],op^+,op^-}(P),op^+}$ and M equals the least fixed point of $T_{GL^a_{M,[\![\cdot]\!],op^+,op^-}(P),op^+}$ so that $N \subseteq M$. □

3 Languages Accepted by Finite Automaton

In this section, we shall briefly list some of the basic properties of languages accepted by finite automaton that we shall need.

Recall that a deterministic finite automaton (DFA) M is specified by a quintuple $M = (Q, \Sigma, \delta, s, F)$ where

Q is a finite alphabet of state symbols,
Σ is finite alphabet of input symbols,
$\delta : Q \times \Sigma \to Q$ is a transition function,
s in Q is the start state, and
$F \subseteq Q$ is the set of final states.

We let $L(M)$ denote the set of all words w accepted by M. A nondeterministic automaton (NFA) $M = (Q, \Sigma, \delta, s, F)$ is specified by similar 5-tuple except that in this case $\delta \subseteq Q \times \Sigma \times Q$. It is well known that for any fixed finite alphabet Σ, the set of languages $L \subseteq \Sigma^*$ accepted by DFA's and the set languages of $L \subseteq \Sigma^*$ accepted by NFA's are the same. Moreover, given any two DFA's M_1 and M_2, there are standard constructions of DFA's M_3, M_4, and M_5 such that

$L(M_3) = L(M_1) \cap L(M_2)$,
$L(M_4) = L(M_1) \cup L(M_2)$, and
$L(M_5) = \Sigma^* - L(M_1)$.

We shall denote these three DFA's by $M_3 = M_1 \cap M_2$, $M_4 = M_1 \cup M_2$, and $M_5 = \bar{M_1}$

A crucial property of DFA's is the pumping lemma.

Lemma 1. *Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA and $p = |Q|$. Then for all words $w \in L(M)$ such that $|w| \geq p$, we can write $w = xyz$ for some $x, y, z \in \Sigma^*$ such that*

1. *$|xy| \leq p$,*
2. *$|y| \geq 1$, and*
3. *$xy^i z \in L(M)$ for all $i \geq 0$.*

One immediate consequence of the pumping lemma is that we can effectively decide whether $L(M)$ is empty or finite. That is, we have the following lemmas.

Lemma 2. *Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA. Then, $L(M)$ is empty if and only if for every $w \in \Sigma^*$ such that $|w| < |Q|$, w is not accepted by $L(M)$.*

Lemma 3. *Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA. Then, $L(M)$ is finite if and only if for every $w \in \Sigma^*$ such that $|Q| \leq |w| < 2|Q|$, w is not accepted by $L(M)$.*

Thus the complexity of the decision procedure to decide whether $L(M)$ is empty or finite depends directly on $|Q|$ and $|\Sigma|$. The fact that we can effectively decide if $L(M) = \emptyset$ also means that we can decide for any given DFA's M_1 and M_2 whether

1. $L(M_1) \subseteq L(M_2)$ since $L(M_1) \subseteq L(M_2)$ if and only if $L(M_1 \cap \bar{M_2}) = \emptyset$,
2. $L(M_1) = L(M_2)$ since $L(M_1) = L(M_2)$ if and only if $L(M_1) \subseteq L(M_2)$ and $L(M_2) \subseteq L(M_1)$, and
3. $L(M_1) \cap L(M_2) = \emptyset$ since $L(M_1) \cap L(M_2) = \emptyset$ if and only if $L(M_1 \cap M_2) = \emptyset$.

4 Set Based Logic Programming with Automata

In this section, we shall consider finite set based logic programs P over $\mathcal{L} = (L, X, [\![\cdot]\!])$ where $X = \Sigma^*$ for some finite alphabet Σ. Thus P consists of clauses of the form

$$\mathcal{C} = A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, C_m \tag{4}$$

where $A, B_1, \ldots, B_n, C_1, \ldots, C_m$ are atoms. We shall assume that $X = \Sigma^*$ for some finite alphabet Σ and that for any clause of the form (4) in P,

$$[\![A]\!], [\![B_1]\!], \ldots, [\![B_n]\!], [\![C_1]\!], \ldots, [\![C_m]\!]$$

are all accepted by DFA's whose alphabet of symbols is Σ. **For the moment, assume also that op^+ and op^- are the identity operators.** For ease of notation, we shall assume that for any atom A that appears in P, A is a DFA whose over the alphabet Σ and that $[\![A]\!] = L(A)$.

Proposition 1. *For every finite set based program P where $op^+ = op_{id}$, every weak or strong stable model of P is a finite union of the sense assignments of the heads of clauses in P.*

Thus any weak or strong stable model of P must be a finite union of languages in Σ^* which are accepted by DFA's and, hence, the stable model itself is accepted by a DFA since languages accepted by DFA's are closed under union. We claim that if M is a DFA whose alphabet of symbols is Σ, then we can effectively decide whether $L(M)$ is a weak or strong stable model of P.

The first thing to observe is that we can effectively find the weak or strong Gelfond-Lifschitz transform of P. That is, under our assumptions for any atom A and any $a \in \{s, w\}$,

1. $L(M) \models^a_{[\![\cdot]\!], op^+, op^-} A$ if and only if $L(A) \subseteq L(M)$,
2. $L(M) \models^s_{[\![\cdot]\!], op^+, op^-} \neg A$ if and only if $L(A) \cap L(M) = \emptyset$, and
3. $L(M) \models^w_{[\![\cdot]\!], op^+, op^-} \neg A$ if and only if $L(A) \not\subseteq L(M)$.

It follows from the results in Section 3, that we can effectively decide whether $L(M) \models^a_{[\![\cdot]\!], op^+, op^-} A$, $L(M) \models^s_{[\![\cdot]\!], op^+, op^-} \neg A$, and $L(M) \models^w_{[\![\cdot]\!], op^+, op^-} \neg A$. Hence, we can effectively construct $GL^s_{L(M), [\![\cdot]\!], op^+, op^-}(P)$ and $GL^w_{L(M), [\![\cdot]\!], op^+, op^-}(P)$.

Now suppose that Q is a finite Horn set based logic program over $\mathcal{L} = (L, X, [\![\cdot]\!])$ where $X = \Sigma^*$ for some finite alphabet Σ and op^+ and op^- are the identity operators. Moreover, assume that for any atom A which appears in Q, $[\![A]\!]$ is a language accepted by a DFA whose alphabet is Σ. Again, for ease of notation, we shall assume that for any atom A that appears in P, A is a DFA whose alphabet is Σ and that $[\![A]\!] = L(A)$. Then, we claim that we can effectively construct a DFA M such that $L(M)$ is the least model of Q. First, we shall show that for all $n \geq 1$, we can effectively construct a DFA M^n such $T^n_{Q, op^+}(\emptyset) = L(M^n)$. Note that $T_{Q, op^+}(\emptyset)$ is equal to $\bigcup\{L(A) : A \leftarrow \ \in Q\}$. Now if $\{A \leftarrow \ \in Q\}$ is empty, then $T_{Q, op^+}(\emptyset) = \emptyset$ and the least model of Q

equals $\emptyset$ so that we simply let M^1 be the one state DFA which has no accepting state. Otherwise, suppose

$$\{A : A \leftarrow \quad \in Q\} = \{A_1^0, \ldots, A_{n_0}^0\}.$$

Then, we set $M^1 = A_1^0 \cup \cdots \cup A_{n_0}^0$. Now assume that we have constructed a DFA M^n such that $T^n_{Q,op^+}(\emptyset) = L(M^n)$. Then,

$$T_{Q,op^+}(L(M^n)) = op^+(I_1 \cup I_2)$$

where $I_1 = \bigcup\{[\![A]\!] \mid A \leftarrow B_1, \ldots, B_m \ \in Q, L(M^n) \models_{[\![\cdot]\!],op^+} B_i, i = 1, \ldots, n\}$ and $I_2 = \bigcup\{[\![A]\!] \mid A \leftarrow$ is a clause in the EDB of $Q\}$.

Note that $I_1 \cup I_2$ is finite since Q is finite. Since we can effectively decide whether $L(N) \subseteq L(M^n)$ for any DFA N, we can effectively decide whether $L(M^n) \models_{[\![\cdot]\!],op^+} B_i$ for any atom B_i and hence we can effectively compute I_1 and I_2. Then we simply let $L(M^{n+1})$ be the DFA whose language is the union of all the $L(A)$ such that $A \in I_1 \cup I_2$.

Finally, we can effectively check whether $L(M^{n+1}) = L(M^n)$. Since the least model of Q equals $L(M^n)$ where n is the least integer such that $L(M^{n+1}) = L(M^n)$, we can effectively construct a DFA R such that $L(R)$ is the least model of Q.

It follows that we can effectively construct DFA's M_s and M_w such that $L(M_s)$ is the least model of $GL^s_{L(M),[\![\cdot]\!],op^+,op^-}(P)$ and $L(M_w)$ is the least model of $GL^w_{L(M),[\![\cdot]\!],op^+,op^-}(P)$. Since we can effectively check whether $L(M) = L(M_s)$ and whether $L(M) = L(M_w)$, it follows that we can effectively decide if $L(M)$ is a weak or strong stable model of P.

We can extend our analysis to finite set based logic programs P with miops assuming that the miops for P satisfy the following property.

Definition 4. We say that a miop $op : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ is *effectively automata preserving* if for any DFA M whose underlying alphabet of symbols is Σ, we can effectively construct a DFA N whose underlying alphabet of symbols is Σ such that $L(N) = op(L(M))$.

We will now give a number of examples of miops on regular languages.

Example 3. Suppose that $\Sigma = \{0, 1, \ldots, m\}$. Then, the following are effectively automata preserving operators.

(1) If N is a DFA whose underlying set of symbols is Σ, then we can define $op : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ by setting $op(S) = S \cup L(N)$ for any $S \subseteq \Sigma^*$. Clearly if $S = L(M)$ for some DFA M whose underlying set of symbols is Σ, then $op(L(M)) = L(M \cup N)$ so op is effectively automaton preserving.
(2)If N is a DFA whose underlying set of symbols is Σ, then we can define $op : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ by setting $op(S) = S \cap L(N)$ for any $S \subseteq \Sigma^*$. Clearly if $S = L(M)$ for some DFA M whose underlying set of symbols is Σ, then $op(L(M) = L(M \cap N)$ so op is effectively automata preserving.

(3)If T is any subset of Σ, we can let $op(S) = ST^*$. Again op will be an effectively automata preserving miop since if M is DFA whose underlying set of symbols is Σ, then let N be NFA constructed from M by adding loops on all the accepting states labeled with letters from T. It is easy to see that N accepts $L(M)T^*$ and then one can use the standard construction to find a DFA N' such that $L(N') = L(N)$. Note that in the special case where T equals Σ, we can think of op as constructing the upper ideal of S in Σ^* relative to the partial order $\sqsubseteq$. That is, we say that for words $u, v \in \Sigma^*$, $u \sqsubseteq v$ if u is prefix of v, i.e. v is of the form uw for some $w \in \Sigma^*$. For any poset $(P, \leq_P)$, we say that a set $U \subseteq P$ is an *upper ideal* in P, if whenever $x \leq_P y$ and $x \in P$, then $y \in P$. Clearly, for the poset $(\Sigma^*, \sqsubseteq)$, $op(S)$ is the upper ideal of $(\Sigma^*, \sqsubseteq)$ generated by S.
(4) Let $\mathcal{P} = (\Sigma, \leq)$ be a poset. For any $w, w' \in \Sigma^*$, we say that w' is a factor of w if there are words $u, v \in \Sigma^*$ with $w = uw'v$. Define the *generalized factor order* on P^* by letting $u \leq w$ if there is a factor w' of w having the same length as u such that $u \leq w'$, where the comparison of u and w' is done componentwise using the partial order in $\mathcal{P}$. Again we can show that if $op(S)$ is the upper ideal generated by S the generalized factor order relative to P^*, then op is an effectively automata preserving miop. That is, if we start with a DFA $M = (Q, \Sigma, \delta, s, F)$, then we can modify M to an NFA that accepts $op(L(M))$ as follows. Think of M as a digraph with edges labeled by elements of Σ in the usual manner. First, we add a new start state s_0. There are loops from s_0 labeled with all letters in Σ. There is also a λ-transition from s_0 to the old start state s. We then modify the transitions in M so that if there is an edge from state q to q' labeled with symbol r, then we add an edge from q to q' with any symbol s such that $r \leq s$. Finally we add loops to all accepting states such that labeled with all letters in in Σ.
(5) If we allow multiple representations of the infinite dimensional vector space V_∞ for the field GF_q where q is prime, then the operator op_{subsp} can be thought of an automaton preserving miop. Let $\Sigma = \{0, \ldots, q-1\}$. The standard way to represent the elements of V_∞ is to let $\mathbf{0} = 0$ and think of a non-zero element of V_∞ as a finite sequence $\sigma_1 \ldots \sigma_n$ where $\sigma_n \neq 0$. The operations of scalar multiplication and addition are then performed componentwise. In our case, we will let any element $\sigma \in V_\infty$ have multiple representations, namely, σ can be represented by $\sigma 0^n$ for any $n \geq 0$. Then, we let $op_{subsp}(S)$ be the set of all representatives of the subspace of V_∞ generated by S. In what follows, we shall only describe how to construct NFA's that accept the desired languages since the Myhill-Nerode Theorem allows us to construct in a uniform manner, for any NFA M, a DFA D such that $L(M) = L(D)$. First, consider miop op_1 such that $op_1(S)$ is the set of all representations of elements of S. If M is a DFA whose underlying alphabet is Σ, then we can modify M to an NFA N that accepts $op_1(S)$ as follows. First, any state q such that there is an n such that the word 0^n starting at state q ends in an accepting state is an accepting state of N. In particular, every accepting state of M is an accepting state of N. In addition, we add loops labeled with 0 to all the accepting states of N.

Next we let $op_2(S)$ denote the set of all representations of any element which is a scalar multiple of an element of S. We claim op_2 is also an automaton preserving miop. That is, if M is a DFA whose underlying alphabet is Σ, then we can modify M to an NFA $\bar{N}$ that accepts $op_2(S)$ as follows. First, let N be the NFA such that $op_1(L(M)) = L(N)$. The for each $a \in \{0, \ldots, q-1\}$, let aN be the NFA that is constructed from N by replacing each edge labeled with the letter x by an edge labeled ax. Then, it is clear that $L(aN) = \{(a\sigma_1)\ldots(a\sigma_n) : \sigma_1 \ldots \sigma_n \in L(N)\}$ so that $op_2(L(M)) = L(\bar{N})$ where $\bar{N} = 0N \cup 1N \cup \cdots \cup (q-1)N$.

Finally for any $a, b \in \{0, \ldots, q-1\}$, we let $op_{a,b}(S)$ denote the set of all representatives of the form $a\sigma + b\tau$ such that σ, τ are in S and $|\sigma| = |\tau|$. $op_{a,b}$ is not a miop, but nevertheless for any DFA M, we can construct an NFA $R_{a,b}$ such that $L(R_{a,b}) = op_{a,b}(L(M))$. First, let $N = (Q, \Sigma, \delta, s, F)$ be the DFA such that $L(N) = op_2(L(M))$. Then, the set of states of $R_{a,b}$ will be $Q \times Q$, (s, s) will be the start state of $R_{a,b}$, and $F \times F$ will be the set of final states of $R_{a,b}$. Now suppose that there are edges from p_0 to p_1 labeled with α and from q_0 to q_1 labeled with β in N. Then, we will have an edge in $R_{a,b}$ from (p_0, q_0) to (p_1, q_1) labeled with $a\alpha + b\beta$. It is easy to see that $L(R_{a,b}) = op_{a,b}(L(M))$. and hence if we let R be the DFA such that $R = \bigcup_{(a,b)\in\Sigma\times\Sigma} R_{a,b}$, then $S \subseteq L(R) \subseteq op_{subsp}(S)$ and $L(R)$ has the property that if $s_1, s_2 \in S$, then $as_1 + bs_2 \in L(R)$ for any $a, b \in GF_q$. By a similar argument, we can construct for any finite sequence of distinct elements $a_1, \ldots, a_r$ from GF_q, a DFA $U_{a_1,\ldots,a_r}$ such that $L(U_{a_1,\ldots,a_r})$ equals the set of all $a_1t_1 + \cdots + a_rt_r$ such that $t_1, \ldots t_r \in L(R)$. It then follows that $op_{subsp}(S)$ equal the union of $L(U_{a_1,\ldots,a_r})$ over all possible finite sequence of distinct elements from GF_q and hence is we can construct a DFA U which accepts $op_{subsp}(S)$.

It is then easy to check that if $op^+ : 2^{\Sigma^*} \to 2^{\Sigma^*}$, then for any Horn set based logic program Q with the properties described above, we can construct a DFA M^n such that $T^n_{Q,op^+}(\emptyset) = L(M^n)$ and, hence, we can effectively construct the least model of Q. Thus we have the following result.

Theorem 3. *Suppose that P is a finite set based logic program over $\mathcal{L} = (L, X, [\![\cdot]\!])$ where $X = \Sigma^*$ for some finite alphabet Σ and $op^+ : 2^{\Sigma^*} \to 2^{\Sigma^*}$ and $op^- : 2^{\Sigma^*} \to 2^{\Sigma^*}$ are effectively automaton preserving miops. Moreover, assume that for any atom A which appears in Q, $[\![A]\!]$ is a language accepted by a DFA whose underlying set of symbols is Σ. Then,*

1. *Every weak (strong) stable model of P is a language accepted by a DFA.*
2. *For any DFA M whose underlying set of symbols is Σ, we can effectively decide whether $L(M)$ is a weak or strong stable model of P.*

Note that under the assumptions of Theorem 3, there are only finitely many possible strong or weak stable models the program P, namely a union of the sense of the head of certain clauses, and these are all recognizable by DFA's. Hence it is decidable whether such a set based logic program has a weak or strong stable model and there is an algorithm to find all such weak or strong stable models.

5 Conclusions

We showed that if the senses of the atoms of a finite set based logic program P are all regular languages over some fixed finite alphabet and the miops involved are all automaton preserving miops, then we can effectively decide if P has weak or strong stable model and there is an algorithm to find all weak and strong stable models. In fact, it is not difficult to see that all the operations in searching for either a weak or strong stable model of such programs are effective so that it is possible to extend existing search engines to produce either weak or strong stable models of such programs. However, we suspect that the problem of how to optimize such extensions of existing search engines will be an interesting and challenging research problem. Finite automaton are useful for carrying out a lot of recognition tasks such as search for keywords or ensuring documents or strings have a proper form so that our results show that we can add ASP programming on top of such recognition tasks.

Finally, we should note that the key properties of DFA's that we used here was that languages accepted by DFA's are closed under union, intersection, and complementation and that we can effectively decide whether the language accepted by a DFA is empty. There are many other classes of automata such a tree automata and Büchi automata that have similar properties so that the results of this paper can easily be extended to cover such classes of automata.

Acknowledgments. The second author has been partially supported by NSF grant DMS 0654060.

References

[AB90] Apt, K., Blair, H.A.: Arithmetic Classification of Perfect Models of Stratified Programs. Fundamenta Informaticae 13, 1–17 (1990)

[BL02] Babovich, Y., Lifschitz, V.: Cmodels (2002), `http://www.cs.utexas.edu/users/tag/cmodels.html`

[Ba03] Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, Cambridge (2003)

[BMR01] Blair, H.A., Marek, V.W., Remmel, J.B.: Spatial Logic Programming. In: Proceedings SCI 2001, Orlando, FL (July 2001)

[BMR08] Blair, H.A., Marek, V.W., Remmel, J.B.: Set Based Logic Programming (in print)

[BMS95] Blair, H.A., Marek, V.W., Schlipf, J.S.: The expressivness of locally stratified programs. Annals of Mathematics and Artificial Intelligence 15, 209–229 (1995)

[BG02] Blumensath, A., Grädel, E.: Automatic Structures. In: Proceedings of the 15^{th} Symposium on Logic in Computer Science LICS 2000, pp. 51–62 (2000)

[Den00] Denecker, M.: Extending classical logic with inductive definitions. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS, vol. 1861, pp. 703–717. Springer, Heidelberg (2000)

[DG82] Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing satisfiability of propositional Horn formulae. Journal of Logic Programming 3, 267–284 (1984)

[GKN+] Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Clasp – a Conflict-driven Answer Set Solver. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS, vol. 4483, pp. 260–265. Springer, Heidelberg (2007)

[GL02] Gelfond, M.: Logic Programming and Knowledge Representation – A-Prolog perspective. Artificial Intelligence Journal 138, 3–38 (2002)

[GL88] Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of the International Joint Conference and Symposium on Logic Programming, pp. 1070–1080. MIT Press, Cambridge (1988)

[GLM06] Giunchiglia, E., Lierer, Y., Maratea, M.: Answer Set Programming Based on Propositional Satisfiability. Journal of Automated Reasoning 36, 345–377 (2006)

[KN95] Khoussainov, B., Nerode, A.: Automatic Presentations of Structures. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 367–392. Springer, Heidelberg (1995)

[KNRS07] Khoussainov, B., Nies, A., Rubin, S., Stephan, F.: Automatic structures: richness and limitations. Logical Methods of Computer Science 3(2), 18 (2007) (electronic)

[LPF+] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Transactions on Computational Logic (2006)

[Li94] Lifschitz, V.: Minimal belief and negation as failure. Artificial Intelligence 70, 53–72 (1994)

[LZ02] Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. In: Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002), pp. 112–117. AAAI Press, Menlo Park (2002)

[MNR94] Marek, W., Nerode, A., Remmel, J.B.: The stable models of predicate logic programs. Journal of Logic Programming 21(3), 129–154 (1994)

[MT93] Marek, W., Truszczyński, M.: Nonmonotonic Logic – Context-Dependent Reasoning. Springer, Heidelberg (1993)

[MT99] Marek, V.W., Truszczyński, M.: Stable Models and an Alternative Logic Programming Paradigm. In: The Logic Programming Paradigm. Series Artificial Intelligence, pp. 375–398. Springer, Heidelberg (1999)

[Nie99] Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25, 3,4, 241–273 (1999)

[SNS02] Simons, P., Niemelä, I., Soininen, T.: Extending and implementing stable semantics of logic programs. Artificial Intelligence 138, 181–234 (2002)

[Sm68] Smullyan, R.: First-order Logic. Springer, Heidelberg (1968)

A Labeled Natural Deduction System for a Fragment of *CTL**

Andrea Masini*, Luca Viganò**, and Marco Volpe***

Department of Computer Science, University of Verona, Italy
{andrea.masini,luca.vigano,marco.volpe}@univr.it

Abstract. We give a sound and complete labeled natural deduction system for an interesting fragment of CTL^*, namely the until-free version of $BCTL^*$. The logic $BCTL^*$ is obtained by referring to a more general semantics than that of CTL^*, where we only require that the set of paths in a model is closed under taking suffixes (i.e. is suffix-closed) and is closed under putting together a finite prefix of one path with the suffix of any other path beginning at the same state where the prefix ends (i.e. is fusion-closed). In other words, this logic does not enjoy the so-called limit-closure property of the standard CTL^* validity semantics.

1 Introduction

The importance of temporal logic in computer science has become clear since the seminal work of Pnueli in 1977 [9]. Interesting applications include its use as a tool for the specification and verification of programs and protocols, in the study and development of temporal databases, as a framework within which to define the semantics of temporal expressions in natural language, and as a language for encoding temporal knowledge in artificial intelligence.

Many branching temporal logics have been proposed in the literature (see [3] for a survey) varying both in the set of the operators used and in the semantics adopted. In particular, the branching-time logic CTL^* (full computation tree logic [5]) has been shown to be especially useful in developing and checking the correctness of reactive systems. In spite of its great relevance, the problem of presenting a satisfactory deduction system or even an Hilbert-style axiomatization for such a logic has been solved only recently in [12].

The aim of this work is to give a sound and complete deduction system for an interesting fragment of CTL^*, namely the until-free version of $BCTL^*$ [14]. The logic $BCTL^*$, which coincides with the logic $\forall LTFC$ described in [16], is obtained by referring to a more general semantics than that of CTL^*, where we only require that the set of paths in a model is closed under taking suffixes

* Partially supported by the PRIN project "CONCERTO".
** Partially supported by the PRIN project "SOFT" and the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu).
*** Partially supported by the PRIN project "SOFT".

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 338–353, 2009.

(i.e. is *suffix-closed*) and is closed under putting together a finite prefix of one path with the suffix of any other path beginning at the same state where the prefix ends (i.e. is *fusion-closed*). In other words, this logic does not enjoy the so-called limit-closure property of the standard CTL^* validity semantics.

The until-free $BCTL^*$ logic that we consider here, to which we give the name $BCTL^*_-$, restricts the set of linear temporal operators to X and G (with the usual intended meanings of "in the next time-instant" and "always in the future" respectively) and includes the universal path quantifier $\forall$, but not the until operator. As in CTL^* (and unlike CTL), we do not constrain temporal operators to be preceded by a path quantifier. From a semantic point of view, we refer to the notion of *bundled validity* (see, e.g., [12]), which will be further clarified in the following.

We give our natural deduction system for $BCTL^*_-$ in the style of *labeled deduction*, a framework [6] that has been successfully employed for several non-classical, and in particular modal, logics [15,19], since labeling provides a clean and effective way of dealing with modalities and gives rise to deduction systems with good proof-theoretical properties. The basic idea is that labels allow one to explicitly encode additional information, of a semantic or proof-theoretical nature, that is otherwise implicit in the logic one wants to capture. So, for instance, instead of a formula A, we consider the *labeled formula* $x : A$, which intuitively means that A holds at the world denoted by x within the underlying Kripke semantics. We can also use labels to specify how worlds are related, e.g. the *relational formula* xRy states that the world y is accessible from x.

It is possible to think of a temporal logic (at least the one we consider) as a modal logic, where modal operators are used to reason on (and the accessibility relation to model) the flow of time. In the light of this consideration, we give here a labeled natural deduction system for $BCTL^*_-$, where labeling allows us to formulate simple and intuitive natural deduction inference rules. We use labels to refer to possible paths rather than to time points and this view leads to a clean deduction system in which each operator (X, G, $\forall$) is seen as a modal operator and is endowed with a proper accessibility relation. Relations between the operators are expressed by means of structural rules that do not involve the operators themselves directly.

We show in this paper that our system is sound and complete, and leave for future work a detailed proof-theoretical analysis of the system (e.g. normalization), as well as the investigation of implementing automatic proof search. Moreover, we are currently working at extending the proposed approach to capture richer and more interesting logics such as full CTL^*, for which this work provides a stepping stone.

We proceed as follows. In Section 2, we give a brief presentation of the syntax and semantics and of an axiomatization of $BCTL^*_-$. In Section 3, we give a labeled natural deduction system for it, which we show in Section 4 to be sound (with respect to the given semantics) and in Section 5 to be complete (with respect to the given axiomatization). We conclude, in Section 6, by comparing with related work and discussing future work.

2 The Bundled Temporal Logic $BCTL^*_-$

We introduce here the logic $BCTL^*_-$, i.e. the until-free fragment of $BCTL^*$.

2.1 Syntax

Definition 1. *Given a set $\mathcal{P}$ of propositional symbols, the set of* well-formed $BCTL^*_-$ formulas *is defined by the grammar*

$$\alpha ::= p \mid \bot \mid \alpha \supset \alpha \mid \mathsf{X}\alpha \mid \mathsf{G}\alpha \mid \forall\alpha\,,$$

where $p \in \mathcal{P}$. The set of atomic formulas *is $\mathcal{P} \cup \{\bot\}$.*

The given syntax uses a minimal set of connectives, operators, and path quantifiers. As usual, we can introduce abbreviations and use, e.g., $\neg$, $\wedge$, $\vee$ for the negation, the conjunction, and the disjunction, respectively. For instance, $\neg\alpha \equiv \alpha \supset \bot$. We can also define other temporal operators, e.g. $\mathsf{F}\alpha \equiv \neg\mathsf{G}\neg\alpha$ to express that α holds sometime in the future, and the existential path quantifier, i.e. $\exists\alpha \equiv \neg\forall\neg\alpha$.

To define a labeled deduction system for the logic $BCTL^*_-$, we extend the language with a set of labels and introduce the notions of labeled formula and relational formula. In the following, we will use the letters $b, c, d, \ldots$ (sometimes subscripted or superscripted) to denote labels, the symbol φ to denote a generic formula (either labeled or relational) and the symbol Γ to denote a set of formulas.

Definition 2. *Let $\mathcal{L}$ be a set of labels and let $b, c \in \mathcal{L}$. If α is a well-formed $BCTL^*_-$ formula, then $b : \alpha$ is a* labeled well-formed formula *(*labeled formula *or* lwff *for short). The set of* relational well-formed formulas *(*relational formulas *or* rwffs *for short) is defined as follows:*

$$\rho ::= b \lhd c \mid b \leqslant c \mid b \bullet c\,.$$

In the rest of the paper, we will assume given a fixed denumerable set $\mathcal{L}$ of labels. Intuitively, in our system, a label is used to refer to a path of a computation. Usually, presentations of branching time logics distinguish between

- *state formulas*, whose main operator is a boolean connective or a path quantifier and which are evaluated with respect to a state, and
- *path formulas*, whose main operator is a linear temporal operator and which are evaluated with respect to a path.

In our case, the intended meaning of an lwff $b : \alpha$ is that

- α holds in the initial state of b when α is a state formula, and that
- α holds in the path b when α is a path formula.

This will be further clarified in Section 2.2, where a semantics given only in terms of paths will be presented.

In the rwffs, we use $\lhd$, $\leqslant$ and $\bullet$ with the following intended meaning:

- $b_1 \leqslant b_2$ states that b_2 is a suffix of b_1, i.e. if $b_1 = s_1, s_2, ...$ then $b_2 = s_i, s_{i+1}, ...$ for some $i \geq 1$;
- $b_1 \lhd b_2$ states that b_2 is the maximal proper suffix of b_1, i.e. if $b_1 = s_1, s_2, s_3, ...$ then $b_2 = s_2, s_3, ...$;
- $b_1 \bullet b_2$ states that b_1 and b_2 share the same initial state, i.e. if $b_1 = s_1, s_2, s_3, ...$ and $b_2 = s'_1, s'_2, s'_3, ...$ then $s_1 = s'_1$.

2.2 Semantics

Several alternative semantics have been proposed for the branching-time logics and some equivalence results have also been showed (see, e.g., [2]). In particular, we can give two main notions of validity: the *full validity* and the *bundled validity*[1] (for a detailed account see [3,12]). If we define a *transition frame* as consisting of a set S of states and of a serial relation $\mathcal{R}$ on S, i.e. a relation such that for every s in S there exists a t in S for which $s\mathcal{R}t$ holds, then the notion of full validity is given by defining the semantics with respect to the set of all the $\mathcal{R}$-generable paths, i.e. of all the ω-sequences $s_1, s_2, ...$ such that $(s_i, s_{i+1}) \in \mathcal{R}$ for all $i \in \mathbb{N}$.

In this work, we refer instead to the notion of *bundled validity*, which determines a smaller set of valid formulas and has been used to define the subset of CTL^* called $\forall LTFC$ in [16] and $BCTL^*$ in [14]. Bundled validity is given by considering a predefined subset P of paths that is required to be (as in *bundled transition frames* of [14]):

1. *suffix-closed*, i.e. if the path $s_0, s_1, s_2...$ is in P then the path $s_1, s_2, ...$ is also in P; and
2. *fusion-closed*, i.e. if $s_1, s_2, ..., s_n, s_{n+1}, s_{n+2}, ...$ and $s'_1, s'_2, ..., s_n, s'_{n+1}, s'_{n+2}, ...$ are in P then $s_1, s_2, ..., s_n, s'_{n+1}, s'_{n+2}, ...$ is also in P.

However, here we prefer to consider a different but equivalent formulation given by frames where the basic entities (or *worlds*, in a Kripke-style terminology) are the paths of computation rather than the states. In fact, this view allows us to present a more genuine Kripke-style semantics, closer to the interpretation we want to give to the set of rules of our system.

We thus introduce $(\mathbb{N} \times \mathcal{W})$-structures [12], which are closely related to the Kamp and Ockhamist structures, described respectively in [17] and [21].

Definition 3. *A* (floored) Ockhamist frame *of countable height (in the following just* Ockhamist frame*) is a triple* $(\mathcal{T}, \prec, \simeq)$ *where:*

1. $\mathcal{T}$ *is the set of points;*
2. $\prec$ *is a transitive, anti-symmetric, irreflexive, linear relation on* $\mathcal{T}$*, i.e.:*
   (a) $\forall x, y, z. ((x \prec y) \wedge (y \prec z)) \Rightarrow (x \prec z)$*;*

[1] An example showing that the full and the bundled validity are distinct notions is given by the formula $\alpha \equiv \forall \mathsf{G}(p \supset \exists \mathsf{X} p) \supset (p \supset \exists \mathsf{G} p)$, where p is an atomic formula. It is possible to check (see [12]) that α is valid with respect to the full semantics but not with respect to the bundled one.

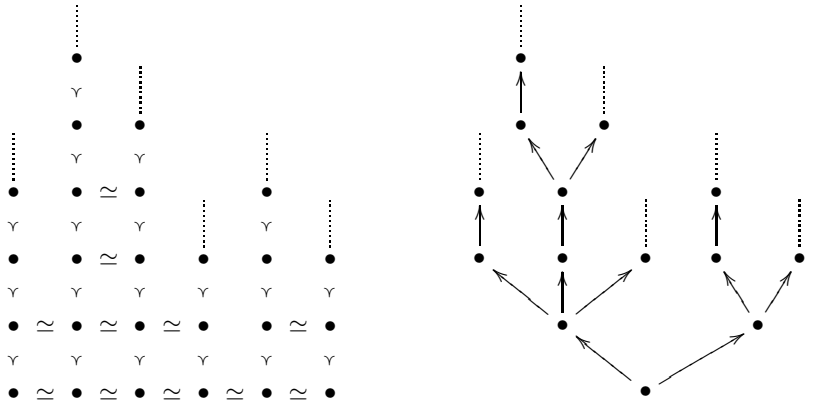


Fig. 1. An *Ockhamist frame* (left) and the corresponding *transition frame* (right)

   (b) $\forall x, y. \neg ((x \prec y) \wedge (y \prec x))$;
   (c) $\forall x. \neg (x \prec x)$;
   (d) $\forall x, y, z. ((x \prec y) \wedge (x \prec z)) \Rightarrow ((z \prec y) \vee (z = y) \vee (y \prec z))$;
   (e) $\forall x, y, z. ((y \prec x) \wedge (z \prec x)) \Rightarrow ((z \prec y) \vee (z = y) \vee (y \prec z))$;
3. $\{y \mid y \prec x\}$ *is finite for each* $x \in \mathcal{T}$;
4. $\simeq$ *is an equivalence relation such that:*
   (a) if $x \simeq y$ *then it is not the case that* $x \prec y$;
   (b) if $x \simeq y$ *and* $u \prec x$ *then there is a* v *such that* $v \prec y$ *and* $u \simeq v$;
5. *there is an element* $0 \in \mathcal{T}$ *such that for each* $w \in \mathcal{T}$, *there is a* $w' \in \mathcal{T}$ *such that* $0 \simeq w'$ *and either* $w' \prec w$ *or* $w' = w$ *(the equivalence class* $0/_{\simeq}$ *is known as the* floor*).*

Intuitively, every Ockhamist point can be thought of as corresponding to a path in a transition frame and the relation $\prec$ as the equivalent of the relation "is a prefix of", i.e. $x \prec y$ stands for "the path x is a prefix of the path y". The branching nature of Ockhamist frames is hidden in the $\simeq$-equivalence relation, where the idea is that each $\simeq$-class of points contains all the paths of the corresponding transition frame that share a same initial state.

More precisely, there exists a translation [13] between Ockhamist frames and bundled transition frames (as exemplified in Fig. 1) based on the fact that Ockhamist points correspond to paths in the transition frame while points related by $\simeq$ correspond to paths with the same initial state.

In order to give a proper semantics for every linear temporal operator, we require the lines of points defined by $\prec$ to be isomorphic to the natural numbers.

Definition 4. *An Ockhamist frame* $(\mathcal{T}, \prec, \simeq)$ *is an* $(\mathbb{N} \times \mathcal{W})$-frame *iff*

1. *there is some set* $\mathcal{W}$ *such that* $\mathcal{T} = (\mathbb{N} \times \mathcal{W})$;
2. *the order* $\prec$ *is defined by* $(n, u) \prec (m, v)$ *iff* $n < m$ *and* $u = v$.

As usual, we obtain a structure by providing the frame a valuation function. In this case, we also need to require that all points in a $\simeq$-equivalence class satisfy the same set of atoms.

Definition 5. *The structure* $(\mathcal{T}, \prec, \simeq, \mathcal{V})$ *is an* $(\mathbb{N} \times \mathcal{W})$-structure *iff* $(\mathcal{T}, \prec, \simeq)$ *is an* $(\mathbb{N} \times \mathcal{W})$*-frame,* $\mathcal{V} : (\mathbb{N} \times \mathcal{W}) \to 2^{\mathcal{P}}$*, and for all* $n \in \mathbb{N}$ *and for all* $u, v \in \mathcal{W}$*, if* $(n, u) \simeq (n, v)$ *then* $\mathcal{V}(n, u) = \mathcal{V}(n, v)$*.*

It is easy to show by induction the following lemma (see [13]), which will be useful later on.

Lemma 1. *Given an* $(\mathbb{N} \times \mathcal{W})$-structure $(\mathcal{T}, \prec, \simeq, \mathcal{V})$ *and two points* (n, w) *and* (m, v) *in* $\mathcal{T}$*, if* $(n, w) \simeq (m, v)$ *then* $n = m$*.*

In order to give a semantics for our labeled system, we need to define explicitly an interpretation of labels as worlds.

Definition 6. *Given the set of labels* $\mathcal{L}$ *and an* $(\mathbb{N} \times \mathcal{W})$*-structure* $\mathcal{M} = (\mathcal{T}, \prec, \simeq, \mathcal{V})$*, where* $\mathcal{T} = (\mathbb{N} \times \mathcal{W})$ *for some set* $\mathcal{W}$*, an* interpretation *is a function* $\lambda : \mathcal{L} \to \mathcal{T}$ *that maps every label in* $\mathcal{L}$ *to a point in* $\mathcal{T}$*.*

We can now give the notion of truth directly for labeled and relational formulas. Note that truth is defined by having the temporal operators X and G operate along the $\prec$-lines of points, and the quantifier $\forall$ within a $\simeq$-equivalence class.

Definition 7. *Given an* $(\mathbb{N} \times \mathcal{W})$*-structure* $\mathcal{M}{=}(\mathcal{T}, \prec, \simeq, \mathcal{V})$*, where* $\mathcal{T}{=}(\mathbb{N} \times \mathcal{W})$ *for some set* $\mathcal{W}$*, and an interpretation* λ *on it,* truth *for an rwff or lwff* φ *is the relation* $\models^{\mathcal{M},\lambda}$ *defined as follows:*

$$
\begin{array}{lll}
\not\models^{\mathcal{M},\lambda} b : \bot; & & \\
\models^{\mathcal{M},\lambda} b_1 \lhd b_2 & \textit{iff} & \textit{there exist } n \in \mathbb{N} \textit{ and } w \in \mathcal{W} \textit{ such that} \\
 & & \lambda(b_1) = (n, w) \textit{ and } \lambda(b_2) = (n+1, w); \\
\models^{\mathcal{M},\lambda} b_1 \leqslant b_2 & \textit{iff} & \lambda(b_1) = \lambda(b_2) \textit{ or } \lambda(b_1) \prec \lambda(b_2); \\
\models^{\mathcal{M},\lambda} b_1 \bullet b_2 & \textit{iff} & \lambda(b_1) \simeq \lambda(b_2); \\
 & & \\
\models^{\mathcal{M},\lambda} b : p & \textit{iff} & p \in \mathcal{V}(\lambda(b)); \\
\models^{\mathcal{M},\lambda} b : \alpha \supset \beta & \textit{iff} & \models^{\mathcal{M},\lambda} b : \alpha \textit{ implies } \models^{\mathcal{M},\lambda} b : \beta; \\
\models^{\mathcal{M},\lambda} b : \mathsf{X}\alpha & \textit{iff} & \textit{for all } b', \models^{\mathcal{M},\lambda} b \lhd b' \textit{ implies } \models^{\mathcal{M},\lambda} b' : \alpha; \\
\models^{\mathcal{M},\lambda} b : \mathsf{G}\alpha & \textit{iff} & \textit{for all } b', \models^{\mathcal{M},\lambda} b \leqslant b' \textit{ implies } \models^{\mathcal{M},\lambda} b' : \alpha; \\
\models^{\mathcal{M},\lambda} b : \forall\alpha & \textit{iff} & \textit{for all } b', \models^{\mathcal{M},\lambda} b \bullet b' \textit{ implies } \models^{\mathcal{M},\lambda} b' : \alpha.
\end{array}
$$

When $\models^{\mathcal{M},\lambda} \varphi$*, we say that* φ *is* true *in* $\mathcal{M}$ *according to* λ*. By extension:*

$$
\begin{array}{lll}
\models^{\mathcal{M},\lambda} \Gamma & \textit{iff} & \models^{\mathcal{M},\lambda} \varphi \textit{ for all } \varphi \in \Gamma; \\
\Gamma \models^{\mathcal{M},\lambda} \varphi & \textit{iff} & \models^{\mathcal{M},\lambda} \Gamma \textit{ implies } \models^{\mathcal{M},\lambda} \varphi; \\
\models^{\mathcal{M}} \varphi & \textit{iff} & \textit{for every interpretation } \lambda, \models^{\mathcal{M},\lambda} \varphi; \\
\models^{\mathcal{M}} \Gamma & \textit{iff} & \textit{for every interpretation } \lambda, \models^{\mathcal{M},\lambda} \Gamma; \\
\Gamma \models \varphi & \textit{iff} & \textit{for every } (\mathbb{N} \times \mathcal{W})\textit{-structure } \mathcal{M} \textit{ and interpretation } \lambda, \\
 & & \Gamma \models^{\mathcal{M},\lambda} \varphi.
\end{array}
$$

We will also write $\models^{\mathcal{M},\lambda(b)} \alpha$ *for* $\models^{\mathcal{M},\lambda} b : \alpha$*, which also illustrates how truth for lwffs is related to the standard truth relation for modal and temporal logics.*

Definition 8. *We call $BCTL^*_-$ the set*

$$\{\alpha \mid \models^{\mathcal{M}} b : \alpha \text{ for every } b \text{ and every } (\mathbb{N} \times \mathcal{W})\text{-structure } \mathcal{M}\}.$$

The main goal of this paper is to provide a sound and complete natural deduction system for $BCTL^*_-$, which we will do in Section 3.

2.3 A Hilbert-Style Axiomatization

We give now a Hilbert-style axiomatization, which we call $\mathcal{H}(BCTL^*_-)$, for the logic $BCTL^*_-$. $\mathcal{H}(BCTL^*_-)$ consists of two sets of axioms (axioms for linear temporal formulas and axioms for quantified formulas) and a set of inference rules. For the first set of axioms, we refer to a standard axiomatization for until-free *LTL* [16]:

$(L1)$ Any tautology instance
$(L2)$ $\mathsf{G}(\alpha \supset \beta) \supset (\mathsf{G}\alpha \supset \mathsf{G}\beta)$
$(L3)$ $(\mathsf{X}\neg\alpha \supset \neg\mathsf{X}\alpha) \wedge (\neg\mathsf{X}\alpha \supset \mathsf{X}\neg\alpha)$
$(L4)$ $\mathsf{X}(\alpha \supset \beta) \supset (\mathsf{X}\alpha \supset \mathsf{X}\beta)$
$(L5)$ $\mathsf{G}\alpha \supset \alpha \wedge \mathsf{XG}\alpha$
$(L6)$ $\mathsf{G}(\alpha \supset \mathsf{X}\alpha) \supset (\alpha \supset \mathsf{G}\alpha)$

The second set of axioms ensures that the path modality $\forall$ behaves as a $\Box$ in the modal logic *S5* and defines some interactions between the linear temporal operators and the path quantifier. This set of axioms comes from [12] and is slightly different from (but clearly equivalent to) the one in [16]:

$(K_\forall)$ $\forall(\alpha \supset \beta) \supset (\forall\alpha \supset \forall\beta)$ $(\forall 1)$ $\forall\alpha \supset \forall\forall\alpha$ $(\forall 2)$ $\forall\alpha \supset \alpha$ $(\forall 3)$ $\alpha \supset \forall\exists\alpha$
$(Atom)$ $p \supset \forall p$ for each atomic proposition p $(Fusion)$ $\forall\mathsf{X}\alpha \supset \mathsf{X}\forall\alpha$

Finally, we have the inference rules of modus ponens and temporal and path generalization:

(MP) If α and $\alpha \supset \beta$ then β
(Nec_X) If α then $\mathsf{X}\alpha$
(Nec_G) If α then $\mathsf{G}\alpha$
$(Nec_\forall)$ If α then $\forall\alpha$

Soundness and completeness of this axiomatization can be easily verified by adapting analogous proofs for similar axiom systems, as in the following lemma.

Lemma 2. *The axiom system $\mathcal{H}(BCTL^*_-)$ is sound and complete for the logic $BCTL^*_-$.*

Proof. (Sketch) The proof mirrors the one given in [16] for $BCTL^*$, with respect to which our axiom system only misses the two axioms concerning the operator *until*, namely:

$(L7)$ $\alpha\mathsf{U}\beta \supset \mathsf{F}\beta$
$(L8)$ $\alpha\mathsf{U}\beta \leftrightarrow \beta \vee (\alpha \wedge \mathsf{X}(\alpha\mathsf{U}\beta))$

where we denote with $\leftrightarrow$ the double implication.[2]

$\mathcal{H}(BCTL^*_-)$ is sound as it is a subset of the axiomatization in [16] and $BCTL^*_-$ structures coincide with $BCTL^*$ structures. A proof of completeness can be easily obtained by adapting the one in [16], which consists of two parts: (i) first a Henkin-style proof is given for the LTL axiomatization, by the definition of a canonical model construction; (ii) then such a construction is extended in order to consider the system for $BCTL^*$. We can modify such a proof for our case by noticing that in (i) the axioms $(L7)$ and $(L8)$ are used along the proof only to deal with formulas containing the operator *until*. We can use the same arguments to show that the axioms $(L1) - (L6)$ form a complete axiomatization for until-free LTL (as it is done for example in [7]). It is also easy to observe that the arguments in (ii) do not make use of the axioms $(L7)$ and $(L8)$. Thus we can mirror part (ii) of the proof in [16] to extend our canonical model construction for until-free LTL to a canonical model construction for $BCTL^*_-$. The main idea here is to consider the equivalence relation between points of the linear canonical model that satisfy the same state formulas and take such equivalence classes as the points of the branching canonical model. $\dashv$

3 The System $\mathcal{N}(BCTL^*_-)$

The only known deduction system for CTL^* is the Hilbert-style axiomatization given in [12]. However, it is a non-standard automata-based axiomatization, which makes use of "an unusual and unorthodox rule of inference" (as stated by Reynolds himself in [14]). Furthermore, if one is interested in a meta-theoretical and proof-theoretical analysis, Hilbert-style axiomatizations are not of a great help. Natural deduction systems have a richer syntactic structure that can be exploited to get interesting meta and proof-theoretical results. In particular, as remarked in the introduction, labeled natural deduction fits in well with the context of modal and temporal logics, by encoding into the syntax semantical properties of such logics.

In this section, we give a labeled natural deduction system, which we call $\mathcal{N}(BCTL^*_-)$, for an interesting fragment of CTL^*, the logic $BCTL^*_-$ described in Section 2. As we observed above, $\mathcal{N}(BCTL^*_-)$ provides a stepping stone for the formalization of a similar system for CTL^* that we are currently working on.

We remark that in the system $\mathcal{N}(BCTL^*_-)$ we do not make use of a proper relational labeling algebra (as, e.g., in [19]) that contains rules that derive rwffs from other rwffs or even lwffs. Since we are mainly interested in the derivation of logical formulas, we rather follow an approach that aims at simplifying the system: we use relational formulas only as side-conditions for the derivation of labeled formulas (as in Simpson's system for intuitionistic modal logic [15]) and thus in $\mathcal{N}(BCTL^*_-)$ there are no rules whose conclusion is a relational formula.

[2] In fact, our set of axioms for the branching part slightly differs from the one in [16] but the two are clearly equivalent, as remarked in [12].

3.1 The Rules of $\mathcal{N}(BCTL^*_-)$

The rules of $\mathcal{N}(BCTL^*_-)$ are given in Fig. 2. There are six kinds of rules, which we describe in the following.

Rules for the Logical Connectives. The rules for the logical connectives mirror those of other labeled natural deduction systems for modal logics [15,19]. $\supset I$ and $\supset E$ are just the labeled version of the standard [10,18] natural deduction rules for implication introduction and elimination, where the notion of *discharged/open assumption* is also standard (e.g. the formula $[b : \alpha]$ is discharged in the rule $\supset I$). The rule $\bot E$ is a labeled version of *reductio ad absurdum*, where we do not enforce Prawitz's side condition that $\alpha \neq \bot$ and we do not constrain the world (b_2) in which we derive a contradiction to be the same (b_1) as in the assumption.

Rules for the Temporal Operators and the Path Quantifier. The rules for the introduction and the elimination of X, G and $\forall$ share the same structure since they all have a "universal" formulation. In fact, let $\Box$ be one of X, G, $\forall$ and let R respectively be one of $\lhd$, $\leqslant$, $\bullet$; the idea is that the meaning of $b_1 : \Box\alpha$ is given by the metalevel implication $b_1 R b_2 \Longrightarrow b_2 : \alpha$ for an arbitrary b_2 R-accessible from b_1 (where the arbitrariness of b_2 is ensured by the side-condition on the introduction rules for X, G and $\forall$).

Rules for $\lhd$. The rule *ser*$\lhd$ models the fact that every world has an immediate successor and thus ensures that the suffix-closure property (as described in Section 2.2) is satisfied. The rule *lin*$\lhd$ specifies that such a successor must be unique.

Rules for $\leqslant$. We recall that $b_1 \leqslant b_2$ intuitively means that b_2 is a suffix of b_1. In terms of the given semantics, $\leqslant$ denotes in the syntax the reflexive and transitive closure of $\prec$ (see Definition 7). The rules *refl* $\leqslant$ and *trans* $\leqslant$ state respectively the reflexivity and transitivity of $\leqslant$.

Rules for $\bullet$. We recall from Section 2.2 that the symbol $\bullet$ in the syntax corresponds to the accessibility relation $\simeq$ in the semantics. $\simeq$ is defined as an equivalence relation and thus we have the rules *refl*$\bullet$, *symm*$\bullet$ and *trans*$\bullet$ that express reflexivity, symmetry and transitivity of $\bullet$ respectively. It follows that $\forall$ behaves as the modal operator $\Box$ does in the modal logic *S5*.

Finally, *atom*$\bullet$ mirrors the property of $(\mathbb{N} \times \mathcal{W})$-structures according to which if $x \simeq y$ then $\mathcal{V}(x) = \mathcal{V}(y)$ (see Definition 5). Intuitively, with regard to transition structures, it models the idea that two paths having the same initial state must satisfy the same set of atomic propositions and is the equivalent of the axiom (*Atom*) in Section 2.3.

Rules for Relations between the Operators. The rule *base* $\leqslant$ expresses the fact that the relation corresponding to $\leqslant$ contains the relation corresponding to

$$\frac{\begin{array}{c}[b_1 : \alpha \supset \bot] \\ \vdots \\ b_2 : \bot\end{array}}{b_1 : \alpha}\ \bot E \qquad \frac{\begin{array}{c}[b : \alpha] \\ \vdots \\ b : \beta\end{array}}{b : \alpha \supset \beta}\ \supset I \qquad \frac{b : \alpha \supset \beta \quad b : \alpha}{b : \beta}\ \supset E$$

$$\frac{\begin{array}{c}[b_1 \lhd b_2] \\ \vdots \\ b_2 : \alpha\end{array}}{b_1 : \mathsf{X}\alpha}\ \mathsf{X}I \qquad \frac{b_1 : \mathsf{X}\alpha \quad b_1 \lhd b_2}{b_2 : \alpha}\ \mathsf{X}E \qquad \frac{\begin{array}{c}[b_1 \lhd b_2] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ ser\lhd \qquad \frac{b_1 \lhd b_2 \quad b_1 \lhd b_3 \quad b_2 : \alpha}{b_3 : \alpha}\ lin\lhd$$

$$\frac{\begin{array}{c}[b_1 \leqslant b_2] \\ \vdots \\ b_2 : \alpha\end{array}}{b_1 : \mathsf{G}\alpha}\ \mathsf{G}I \qquad \frac{b_1 : \mathsf{G}\alpha \quad b_1 \leqslant b_2}{b_2 : \alpha}\ \mathsf{G}E \qquad \frac{\begin{array}{c}[b_1 \leqslant b_1] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ refl\leqslant \qquad \frac{b_1 \leqslant b_2 \quad b_2 \leqslant b_3 \quad \begin{array}{c}[b_1 \leqslant b_3] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ trans\leqslant$$

$$\frac{\begin{array}{c}[b_1 \bullet b_2] \\ \vdots \\ b_2 : \alpha\end{array}}{b_1 : \forall\alpha}\ \forall I \qquad \frac{b_1 : \forall\alpha \quad b_1 \bullet b_2}{b_2 : \alpha}\ \forall E \qquad \frac{\begin{array}{c}[b_1 \bullet b_1] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ refl\bullet \qquad \frac{b_1 \bullet b_2 \quad \begin{array}{c}[b_2 \bullet b_1] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ symm\bullet$$

$$\frac{b_1 \bullet b_2 \quad b_2 \bullet b_3 \quad \begin{array}{c}[b_1 \bullet b_3] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ trans\bullet \qquad \frac{b_1 : p \quad b_1 \bullet b_2}{b_2 : p}\ atom\bullet \qquad \frac{b_1 \lhd b_2 \quad \begin{array}{c}[b_1 \leqslant b_2] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ base\leqslant$$

$$\frac{b_1 \lhd b_2 \quad b_2 \bullet b_3 \quad \begin{array}{c}[b' \bullet b_1] \quad [b' \lhd b_3] \\ \vdots \\ b : \alpha\end{array}}{b : \alpha}\ fusion \qquad \frac{b_0 : \alpha \quad b_0 \leqslant b \quad \begin{array}{c}[b_0 \leqslant b_i] \quad [b_i \lhd b_j] \quad [b_i : \alpha] \\ \vdots \\ b_j : \alpha\end{array}}{b : \alpha}\ ind$$

In $\mathsf{X}I$, b_2 is *fresh*, i.e. it is different from b_1 and does not occur in any assumption on which $b_2 : \alpha$ depends other than the discarded assumption $b_1 \lhd b_2$.
In $ser\lhd$, b_2 is fresh, i.e. it is different from b and does not occur in any assumption on which $b : \alpha$ depends other than the discarded assumption $b_1 \lhd b_2$.
In $\mathsf{G}I$, b_2 is fresh, i.e. it is different from b_1 and does not occur in any assumption on which $b_2 : \alpha$ depends other than the discarded assumption $b_1 \leqslant b_2$.
In $\forall I$, b_2 is fresh, i.e. it is different from b_1 and does not occur in any assumption on which $b_2 : \alpha$ depends other than the discarded assumption $b_1 \bullet b_2$.
In $atom\bullet$, p is an atomic proposition.
In *fusion*, b' is fresh, i.e. it is different from b, b_1, b_2 and b_3, and does not occur in any assumption on which $b : \alpha$ depends other than the discarded assumptions $b' \bullet b_1$ and $b' \lhd b_3$.
In *ind*, b_i and b_j are fresh, i.e. they are different from each other and from b and b_0, and do not occur in any assumption on which $b : \alpha$ depends other than the discarded assumptions of the rule.

Fig. 2. The rules of $\mathcal{N}(BCTL^*_{-})$

$$\dfrac{\dfrac{\dfrac{\dfrac{[b : \mathsf{G}(\alpha \supset \mathsf{X}\alpha)]^1 \quad [b \leqslant d]^4}{d : \alpha \supset \mathsf{X}\alpha}\,\mathsf{G}E \quad [d : \alpha]^4}{d : \mathsf{X}\alpha}\,\supset E \quad [d \lhd d']^4}{d' : \alpha}\,\mathsf{X}E}{\dfrac{[b : \alpha]^2 \quad [b \leqslant c]^3}{c : \alpha}\,ind^4}$$
$$\dfrac{\dfrac{\dfrac{c : \alpha}{b : \mathsf{G}\alpha}\,\mathsf{G}I^3}{b : \alpha \supset \mathsf{G}\alpha}\,\supset I^2}{b : \mathsf{G}(\alpha \supset \mathsf{X}\alpha) \supset (\alpha \supset \mathsf{G}\alpha)}\,\supset I^1$$

$$\dfrac{\dfrac{\dfrac{\dfrac{[b : \forall \mathsf{X}\alpha]^1 \quad [b \bullet b']^4}{b' : \mathsf{X}\alpha}\,\forall E \quad [b' \lhd d]^4}{d : \alpha}\,\mathsf{X}E}{\dfrac{[b \lhd c]^2 \quad [c \bullet d]^3}{d : \alpha}\,fusion^4}$$
$$\dfrac{\dfrac{\dfrac{d : \alpha}{c : \forall\alpha}\,\forall I^3}{b : \mathsf{X}\forall\alpha}\,\mathsf{X}I^2}{b : \forall \mathsf{X}\alpha \supset \mathsf{X}\forall\alpha}\,\supset I^1$$

Fig. 3. Proofs of the $\mathcal{H}(BCTL^*_{\_})$ axioms (*L6*) and (*Fusion*)

$\lhd$: in the "path terminology", it says that every path b is a prefix of its maximal proper suffix.

The rule *fusion* strictly corresponds to the *fusion-closure* property (see Section 2.2) of bundled transition frames, according to which the set of paths must be closed under putting together a finite prefix of one path with the suffix of any other path such that the prefix ends at the same state as the suffix begins. In terms of the given semantics, it roughly corresponds to condition 4(b) in the definition of an Ockhamist frame (Definition 3). In terms of the axiomatization $\mathcal{H}(BCTL^*_{\_})$ in Section 2.3, it is the equivalent of the axiom (*Fusion*).

Finally, we have a rule *ind* modeling the induction principle underlying the relation between $\lhd$ and $\leqslant$. It comes from the definition of $(\mathbb{N} \times \mathcal{W})$-frame (Definition 4), which requires the vertical lines of points to be isomorphic to the natural numbers. The rule is given only in terms of relations between labels, since we prefer (for proof-theoretical reasons) to restrict the treatment of operators in the system to the specific rules for their introduction and elimination.

3.2 Derivations

Given the rules in Fig. 2, the notion of derivation is the standard one for natural deduction systems [10,18]. We write $\Gamma \vdash_{\mathcal{N}(BCTL^*_{\_})} b : \alpha$ to say that there exists a derivation of $b : \alpha$ in the system $\mathcal{N}(BCTL^*_{\_})$ whose open assumptions are all contained in the set of formulas Γ. A derivation of $b : \alpha$ in $\mathcal{N}(BCTL^*_{\_})$ where all the assumptions are discharged is a *proof* of $b : \alpha$ in $\mathcal{N}(BCTL^*_{\_})$ and we then say that $b : \alpha$ is a theorem of $\mathcal{N}(BCTL^*_{\_})$ (and write $\vdash_{\mathcal{N}(BCTL^*_{\_})} b : \alpha$).

As notation, we write

$$\begin{array}{c} \varphi_1 \dots \varphi_n \\ \pi \\ b : \alpha \end{array}$$

to denote that π is a derivation of $b : \alpha$ whose set of assumptions may contain the formulas $\varphi_1, \ldots, \varphi_n$.

As concrete examples, Fig. 3 contains the proofs of the $\mathcal{H}(BCTL^*_-)$ axioms (*L6*) and (*Fusion*).

4 Soundness

Theorem 1. *For every set Γ of labeled and relational formulas and every labeled formula $b : \alpha$, it holds that*

$$\Gamma \vdash_{\mathcal{N}(BCTL^*_-)} b : \alpha \quad \Rightarrow \quad \Gamma \models b : \alpha \, .$$

The proof proceeds by induction on the structure of the derivation of $b : \alpha$. The base case is when $b : \alpha \in \Gamma$ and is trivial. There is one step case for every rule and we show only five representative cases.

Consider an application of the rule $\mathsf{X}I$:

$$\dfrac{\begin{array}{c}[b \lhd b'] \\ \pi \\ b' : \alpha\end{array}}{b : \mathsf{X}\alpha}\ \mathsf{X}I$$

where π is a proof of $b' : \alpha$ from hypotheses in Γ', with b' fresh and with $\Gamma' = \Gamma \cup \{b \lhd b'\}$. By the induction hypothesis, for all interpretations λ, if $\models^{\mathcal{M},\lambda} \Gamma'$ then $\models^{\mathcal{M},\lambda} b' : \alpha$. We let λ be any interpretation such that $\models^{\mathcal{M},\lambda} \Gamma$, and show that $\models^{\mathcal{M},\lambda} b : \mathsf{X}\alpha$. Let (n, w) be any point such that $\lambda(b) = (n, w)$. Since λ can be trivially extended to another interpretation (still called λ for simplicity) by setting $\lambda(b') = (n + 1, w)$, the induction hypothesis yields $\models^{\mathcal{M},\lambda} b' : \alpha$, i.e. $\models^{\mathcal{M},(n+1,w)} \alpha$, and thus $\models^{\mathcal{M},\lambda} b : \mathsf{X}\alpha$.

Consider an application of the rule $\forall I$:

$$\dfrac{\begin{array}{c}[b \bullet b'] \\ \pi \\ b' : \alpha\end{array}}{b : \forall\alpha}\ \forall I$$

where π is a proof of $b' : \alpha$ from hypotheses in Γ', with b' fresh and with $\Gamma' = \Gamma \cup \{b \bullet b'\}$. By the induction hypothesis, for all interpretations λ, if $\models^{\mathcal{M},\lambda} \Gamma'$ then $\models^{\mathcal{M},\lambda} b' : \alpha$. We let λ be any interpretation such that $\models^{\mathcal{M},\lambda} \Gamma$, and show that $\models^{\mathcal{M},\lambda} b : \forall\alpha$. Let (n, w) be any point such that $\lambda(b) = (n, w)$. Now let us consider an arbitrary point (n, w') for some w'. Since λ can be trivially extended to another interpretation (still called λ for simplicity) by setting $\lambda(b') = (n, w')$, the induction hypothesis yields $\models^{\mathcal{M},\lambda} b' : \alpha$, i.e. $\models^{\mathcal{M},(n,w')} \alpha$. Given that w' is arbitrary we can conclude $\models^{\mathcal{M},\lambda} b : \forall\alpha$.

Consider the case in which the last rule applied is $\mathsf{G}E$:

$$\dfrac{\begin{array}{c}\pi \\ b' : \mathsf{G}\alpha \quad b' \leqslant b\end{array}}{b : \alpha}\ \mathsf{G}E$$

where π is a proof of $b' : \mathsf{G}\alpha$ from hypotheses in Γ_1, with $\Gamma = \Gamma_1 \cup \{b' \leqslant b\}$ for some set Γ_1 of formulas. By applying the induction hypothesis on π, we have:

$$\Gamma_1 \models b' : \mathsf{G}\alpha \ .$$

We proceed by considering a generic $(\mathbb{N} \times \mathcal{W})$-structure $\mathcal{M} = (\mathcal{T}, \prec, \simeq, \mathcal{V})$ and a generic interpretation λ on it such that $\models^{\mathcal{M},\lambda} \Gamma$ and showing that this entails

$$\models^{\mathcal{M},\lambda} b : \alpha \ .$$

Since $\Gamma_1 \subset \Gamma$, from the induction hypothesis we deduce $\models^{\mathcal{M},\lambda} b' : \mathsf{G}\alpha$. Furthermore $\models^{\mathcal{M},\lambda} \Gamma$ entails $\models^{\mathcal{M},\lambda} b' \leqslant b$. Then, by Definition 7, we obtain $\models^{\mathcal{M},\lambda} b : \alpha$.

Let an application of *fusion* be the last rule application in the derivation of $b : \alpha$:

$$\dfrac{b_1 \lhd b_2 \quad b_2 \bullet b_3 \qquad \overset{[b' \bullet b_1] \quad [b' \lhd b_3]}{\overset{\pi}{b : \alpha}}}{b : \alpha}\ \textit{fusion}$$

where π is a proof of $b : \alpha$ from hypotheses in Γ_2, with $\Gamma = \Gamma_1 \cup \{b_1 \lhd b_2\} \cup \{b_2 \bullet b_3\}$ and $\Gamma_2 = \Gamma_1 \cup \{b' \bullet b_1\} \cup \{b' \lhd b_3\}$ for some set Γ_1 of formulas. The side-condition ensures that b' is fresh in π. Hence, by applying the induction hypothesis on π, we have

$$\Gamma_2 \models b : \alpha \ .$$

We proceed by considering a generic $(\mathbb{N} \times \mathcal{W})$-structure $\mathcal{M} = (\mathcal{T}, \prec, \simeq, \mathcal{V})$ and a generic interpretation λ on it such that $\models^{\mathcal{M},\lambda} \Gamma$ and showing that this entails

$$\models^{\mathcal{M},\lambda} b : \alpha \ .$$

From $\models^{\mathcal{M},\lambda} \Gamma$, we deduce:

(i) there exists a point $(n, w) \in \mathcal{T}$ such that $\lambda(b_1) = (n, w)$ and $\lambda(b_2) = (n + 1, w)$;
(ii) $\lambda(b_2) \simeq \lambda(b_3)$.

We know from Lemma 1 that $\lambda(b_3) = (n + 1, v)$ for some $(n + 1, v) \in \mathcal{T}$. Then by the property $4(b)$ of Ockhamist frames (Definition 3), the point (n, v) is such that $(n, v) \simeq (n, w) = \lambda(b_1)$. Now let us consider an interpretation λ' which differs from λ only for the point assigned to b', namely $\lambda' = \lambda[b' \mapsto (n, v)]$. Note that we have defined λ' in a way such that $\models^{\mathcal{M},\lambda'} b' \bullet b_1$ and $\models^{\mathcal{M},\lambda'} b' \lhd b_3$. Since b' does not occur in Γ (by the side-condition on the application of *fusion*), we have $\models^{\mathcal{M},\lambda'} \Gamma_1$ and thus also $\models^{\mathcal{M},\lambda'} \Gamma_2$. Then, by the induction hypothesis, $\models^{\mathcal{M},\lambda'} b : \alpha$. We conclude $\models^{\mathcal{M},\lambda} b : \alpha$ by observing that the side-condition $b' \neq b$ ensures $\lambda(b) = \lambda'(b)$.

Finally, consider the case in which the last rule applied is *ind*:

$$\dfrac{\overset{\pi'}{b_0 : \alpha} \quad b_0 \leqslant b \qquad \overset{[b_0 \leqslant b_i] \quad [b_i \lhd b_j] \quad [b_i : \alpha]}{\overset{\pi}{b_j : \alpha}}}{b : \alpha}\ \textit{ind}$$

where π is a proof of $b_j : \alpha$ from hypotheses in Γ_2 and π' is a proof of $b_0 : \alpha$ from hypotheses in Γ_1, with $\Gamma = \Gamma_1 \cup \{b_0 \leqslant b\}$ and $\Gamma_2 = \Gamma_1 \cup \{b_0 \leqslant b_i\} \cup \{b_i \lhd b_j\} \cup \{b_i : \alpha\}$ for some set Γ_1 of formulas. The side-condition on *ind* ensures that b_i and b_j are fresh in π. Hence, by applying the induction hypothesis on π and π', we have:

$$\Gamma_2 \models b_j : \alpha \qquad \text{and} \qquad \Gamma_1 \models b_0 : \alpha \,.$$

We proceed by considering a generic $(\mathbb{N} \times \mathcal{W})$-structure $\mathcal{M} = (\mathcal{T}, \prec, \simeq, \mathcal{V})$ and a generic interpretation λ on it such that $\models^{\mathcal{M},\lambda} \Gamma$ and showing that this entails

$$\models^{\mathcal{M},\lambda} b : \alpha \,.$$

First, we note that $\Gamma_1 \subset \Gamma$ and therefore $\models^{\mathcal{M},\lambda} \Gamma$ implies $\models^{\mathcal{M},\lambda} \Gamma_1$ and, by the induction hypothesis on π', $\models^{\mathcal{M},\lambda} b_0 : \alpha$. Let $\lambda(b_0) = (n, w)$ for some $(n, w) \in \mathcal{T}$. From $\models^{\mathcal{M},\lambda} \Gamma$, we deduce $\models^{\mathcal{M},\lambda} b_0 \leqslant b$ and thus $\lambda(b) = (n + k, w)$ for some $k \in \mathbb{N}$. We show by induction on k that $\models^{\mathcal{M},\lambda} b : \alpha$. As a base case, we have $k = 0$; it follows that $\lambda(b) = \lambda(b_0)$ and thus trivially that $\models^{\mathcal{M},\lambda} b_0 : \alpha$ entails $\models^{\mathcal{M},\lambda} b : \alpha$. Let us consider now the induction step. Given a label b_{k-1} such that $\lambda(b_{k-1}) = (n + k - 1, w)$, we show that the induction hypothesis $\models^{\mathcal{M},\lambda} b_{k-1} : \alpha$ entails the thesis $\models^{\mathcal{M},\lambda} b : \alpha$. We can build an interpretation λ' that differs from λ only in the points assigned to b_i and b_j, namely $\lambda' = \lambda[b_i \mapsto (n+k-1, w)][b_j \mapsto (n+k, w)]$. It is easy to verify that the interpretation λ' is such that the following three conditions hold:

(i) $\models^{\mathcal{M},\lambda'} b_i : \alpha$;
(ii) $\models^{\mathcal{M},\lambda'} b_0 \leqslant b_i$;
(iii) $\models^{\mathcal{M},\lambda'} b_i \lhd b_j$.

Furthermore, the side-condition on the rule *ind* ensures that λ and λ' agree on all the labels occurring in Γ_1, from which we can infer that also $\models^{\mathcal{M},\lambda'} \Gamma_1$ must hold. It follows that $\models^{\mathcal{M},\lambda'} \Gamma_2$ and thus, by the induction hypothesis on π, that $\models^{\mathcal{M},\lambda'} b_j : \alpha$. We conclude $\models^{\mathcal{M},\lambda} b : \alpha$ by observing that $\lambda'(b_j) = \lambda(b)$.

5 Completeness

The proposed natural deduction system $\mathcal{N}(BCTL^*_-)$ consists of only finitary rules; consequently, it cannot be strongly complete.[3] In fact, it is easy to check that $\{b : \mathsf{X}^i\alpha\}_{i<\omega} \models b : \mathsf{G}\alpha$ but (via soundness) we can see that $\{b : \mathsf{X}^i\alpha\}_{i<\omega} \nvdash b : \mathsf{G}\alpha$, where $\mathsf{X}^0\alpha$ is just α and $\mathsf{X}^{i+1}\alpha$ stands for $\mathsf{X}\mathsf{X}^i\alpha$. Nevertheless, our system $\mathcal{N}(BCTL^*_-)$ is weakly complete with respect to $BCTL^*_-$, namely:

Theorem 2. *For every labeled formula $b : \alpha$ it holds:*

$$\models b : \alpha \quad \Rightarrow \quad \vdash_{\mathcal{N}(BCTL^*_-)} b : \alpha \,.$$

[3] This is not a problem of our formulation: all the finitary deduction systems for temporal logics equipped with at least the operators X and G have such a defect (see, e.g., [8, Chapter 6]).

The most "economic" way to prove the theorem is to show that $\mathcal{N}(BCTL^*_-)$ is complete with respect to the axiomatization $\mathcal{H}(BCTL^*_-)$ given in Section 2.3, which is sound and complete for the logic $BCTL^*_-$. That is, we need to prove (i) that every axiom of $\mathcal{H}(BCTL^*_-)$ is provable in $\mathcal{N}(BCTL^*_-)$ and (ii) that $\mathcal{N}(BCTL^*_-)$ is closed under the (labeled equivalent of the) rules of inference of $\mathcal{H}(BCTL^*_-)$. Showing (ii) is straightforward and we omit it here. As for (i), we have already given the proofs of the $\mathcal{H}(BCTL^*_-)$ axioms $(L6)$ and $(Fusion)$ in Fig. 3. As a further example, we can prove axiom $(L5)$ as follows

$$\dfrac{\dfrac{[b : \mathsf{G}\alpha]^1 \quad [b \leqslant b]^2}{\dfrac{b : \alpha}{b : \alpha}\ \mathit{refl} \leqslant^2}\ \mathsf{G}E \qquad \dfrac{\dfrac{[b \lhd c]^3}{\dfrac{\dfrac{[b \leqslant c]^5 \quad [c \leqslant d]^4}{d : \alpha}\ \mathit{trans} \leqslant^6 \quad \dfrac{[b : \mathsf{G}\alpha]^1 \quad [b \leqslant d]^6}{d : \alpha}\ \mathsf{G}E}{d : \alpha}\ \mathit{base} \leqslant^5}{\dfrac{d : \alpha}{c : \mathsf{G}\alpha}\ \mathsf{G}I^4}{b : \mathsf{X}\mathsf{G}\alpha}\ \mathsf{X}I^3}{b : \alpha \wedge \mathsf{X}\mathsf{G}\alpha}\ \wedge I}{b : \mathsf{G}\alpha \supset (\alpha \wedge \mathsf{X}\mathsf{G}\alpha)}\ \supset I^1$$

where, for simplicity, we have employed the rule $\wedge I$ for conjunction introduction, which is derived from the other propositional rules as is standard:

$$\dfrac{b : \alpha_1 \quad b : \alpha_2}{b : \alpha_1 \wedge \alpha_2}\ \wedge I \qquad \text{abbreviates} \qquad \dfrac{\dfrac{[b : \alpha_1 \supset (\alpha_2 \supset \bot)]^1 \quad b : \alpha_1}{b : \alpha_2 \supset \bot}\ \supset E \quad b : \alpha_2}{b : \bot}\ \supset E}{b : (\alpha_1 \supset (\alpha_2 \supset \bot)) \supset \bot}\ \supset I^1$$

6 Conclusions

We have given a labeled natural deduction system for a fragment of CTL^* — $BCTL^*$ without *until* — and shown that it is sound and complete.

We have already considered some relevant related works in the previous sections. Other labeled natural deduction systems for branching time logics have been proposed, e.g. [1] and [11] both give labeled natural deduction systems for CTL. The main distinctive feature of our system is that reasoning only in terms of paths gives us the possibility of considering also the path quantifier $\forall$ as a modal operator and thus of getting a labeled system as clean as the ones for other modal logics [15,19].

In [14], a tableau-based decision procedure for $BCTL^*$ is given. The tableau construction differs from the traditional tree-shaped one and consists, like for other tableau systems for temporal logics, e.g. [4,20], in starting with a graph and iteratively pruning away some nodes until a success or a failure condition is reached. We remark that the focus of our work, instead, mainly concerns the definition of a deduction system with good proof-theoretical properties.

In fact, we are currently working on normalization for our system $\mathcal{N}(BCTL^*_-)$, where the main difficulties arise, as in deduction systems for Peano Arithmetics and as expectable, from the presence of a temporal induction principle. Further

current work is oriented towards automated reasoning and towards extensions of the system in order to capture richer logics such as CTL^*.

References

1. Bolotov, A., Grigoriev, O., Shangin, V.: Natural Deduction Calculus for Computation Tree Logic. In: Proceedings of the John Vincent Atanasoff Symposium, pp. 175–183 (2006)
2. Emerson, E.A.: Alternative Semantics for Temporal Logics. Theoretical Computer Science 26, 121–130 (1983)
3. Emerson, E.A.: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science. Formal Models and Semantics (B), vol. B, pp. 995–1072 (1990)
4. Emerson, E.A., Halpern, J.Y.: Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. Journal of Computer and System Sciences 30(1), 1–24 (1985)
5. Emerson, E.A., Sistla, A.P.: Deciding Full Branching Time Logic. Information and Control 61(3), 175–201 (1984)
6. Gabbay, D.: Labelled Deductive Systems. Clarendon Press (1996)
7. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the Temporal Analysis of Fairness. In: Proceedings of POPL 1980, pp. 163–173 (1980)
8. Kröger, F.: Temporal Logic of Programs. Springer, Heidelberg (1987)
9. Pnueli, A.: The Temporal Logic of Programs. In: Proceedings of FOCS, vol. 18, pp. 46–57 (1977)
10. Prawitz, D.: Natural Deduction. Almqvist and Wiksell (1965)
11. Renteria, C., Haeusler, E.: A Natural Deduction System for CTL. Bulletin of the Section of Logic 31(4), 231–240 (2002)
12. Reynolds, M.: An Axiomatization of Full Computation Tree Logic. Journal of Symbolic Logic 66(3), 1011–1057 (2001)
13. Reynolds, M.: An Axiomatization of PCTL*. Information and Computation 201(1), 72–119 (2005)
14. Reynolds, M.: A Tableau for Bundled CTL*. Journal of Logic and Computation 17(1), 117–132 (2007)
15. Simpson, A.: The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D thesis, College of Science and Engineering, School of Informatics, University of Edinburgh (1994)
16. Stirling, C.: Modal and Temporal Logics. In: Handbook of Logic in Computer Science, vol. 2, pp. 477–563. Oxford University Press, Oxford (1992)
17. Thomason, R.H.: Combinations of Tense and Modality. In: Handbook of Philosophical Logic: Extensions of Classical Logic, pp. 135–165. Reidel (1984)
18. Troelstra, A., Schwichtenberg, H.: Basic Proof Theory. Cambridge University Press, Cambridge (2000)
19. Viganò, L.: Labelled Non-Classical Logics. Kluwer Academic Publishers, Dordrecht (2000)
20. Wolper, P.: The Tableau Method for Temporal Logic: An Overview. Logique et Analyse 110, 119–136 (1985)
21. Zanardo, A.: Branching-Time Logic with Quantification over Branches: The Point of View of Modal Logic. Journal of Symbolic Logic 61(1), 1–39 (1996)

Conservativity for Logics of Justified Belief

Robert S. Milnikel

Kenyon College, Gambier OH 43022 USA
milnikelr@kenyon.edu
http://www2.kenyon.edu/depts/math/milnikel

Abstract. In [1], Fitting showed that the standard hierarchy of logics of justified knowledge is conservative (e.g. a logic with positive introspection operator ! is conservative over the logic without !). We do the same with most logics of justified belief, but taking a semantic approach rather than Fitting's syntactic one. A brief example shows that conservativity does not hold for logics of justified consistent belief.

1 Introduction

In [1], Fitting showed conservativity of logics of justified knowledge, including **JT**, **JT4** (also known as **LP**), and **JT45** as well as many weaker logics. His proof showed something stronger than simple conservation of validity; he showed that simple omission of symbols missing in the smaller language, carefully done, leads to a line-by-line translation of all proofs in the stronger logic into proofs in the weaker one. He observed that his method did not extend to logics of justified belief (such as **J** and **J4**) and left the question of conservativity in these logics open.

For reasons I'll mention below, what appear to me to be the obvious syntactic approaches to conservativity in logics of justified belief do not work. However, an approach based on extending models of the smaller logic to those of the larger works out nicely. In this short note, I will outline the basic definitions of logics of justified belief and a simple semantics for these. I will then present one conservativity argument in detail (the conservativity of **J4** over **J**) and outline others in broad strokes. After an example showing the lack of conservativity of **JD4** over **JD**, I will close with a few comments on open problems.

2 Preliminaries

Modal logic has long been a way of attempting to formalize the idea of knowledge and belief (among other concepts). What separates knowledge from "mere" belief is the truth of what is known, reflected in the modal axiom scheme **T** ($\Box F \rightarrow F$). I will assume the reader is familiar with the modal logics of belief (the basic normal modal logic **K** with or without the assumption of positive introspection $\Box F \rightarrow \Box\Box F$). I will deal later with the deontic axiom **D** and its explicit counterpart, which insist on the consistency of belief.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 354–364, 2009.

In a series of papers ([2], [3], [4] and others), Sergei Artemov defined the Logic of Proofs (**LP**), partially as a solution to an open problem dating back to Gödel ([5]) regarding the proper interpretation of the **S4** modality as arithmetic proof. However, **LP** proved useful and interesting well beyond answering Gödel's question, as a general way to make reasoning about knowledge explicit. Many variations of Artemov's original **LP** have appeared over the last decade or so, and have come under the common heading of *justification logics*.

Why "justification" logics? Because in each of these systems, we augment propositional logic with *justification terms* which are intended to make explicit the reasons for knowing/believing a particular proposition. In a formula $t : F$, the justification term t makes explicit the reasons for asserting that formula F (which may itself contain nested justification terms) is known/believed.

I will limit myself to fairly standard justification logics without truth axioms (the analog of the modal axiom **T**), and will briefly define languages, axiom systems, and a simple semantics before moving on to the main result of the paper. The exposition will be quite limited, directed at setting up terminology and notation for the main proof.

2.1 Languages

As implied several times above, we will be examining several logics of justified belief, which will differ in the richness of their language. All logics of justified belief contain *justification terms*, which include *justification variables* $x_1, x_2, \ldots$ and *justification constants* $c_1, c_2, \ldots$. In addition, a particular language may contain one or more of the following symbols for operations on justification terms:

- $\cdot$ (binary)
- $+$ (binary)
- ! (unary)

The intended meanings of these symbols are as follows:

- $\cdot$ is known as "application," the idea being that if s is a justification for believing $F \rightarrow G$ and t is a justification for believing F, then $s \cdot t$ is a justification for believing G;
- $+$ is known as "sum," sort of a concatenation of justifications, the idea being that $s + t$ is justification for believing anything justified either by s or by t;
- ! is used to represent positive introspection, so that if t is the justification for believing F, then $!t$ is the justification for believing that t is justification for believing F.

We will limit ourselves to $\cdot$, $+$, and !, since the negative introspection operator ? is a recent addition to the literature and the history of results involving the other operators is substantially richer; in addition, negative introspection presents difficulties for reasons mentioned in the concluding section of this paper.

We will define *formulas* as being built from atomic propositions $P_1, P_2, \ldots$, the propositional constant $\bot$ and the implication operator $\rightarrow$ in the usual way; in

addition, given any formula F and justification term t, we will allow the formula $t{:}F$.

Following Fitting ([1]), as I will for much of the following section, I will adopt the notation $L(S)$ (where S is a subset of $\{\cdot, +, !\}$) to indicate the language where only justification operations from S are permitted, and $B(S)$ to indicate the logic of justified belief based on $L(S)$.

2.2 Logics, Axiomatically

Of course languages with different sets of justification operators will have different collections of axioms to govern the behavior of those operators.

- All justification logics include a propositionally complete set of classical axiom schemes.
- If $\cdot \in S$, include the axiom scheme $s{:}(F \to G) \to (s{:}F \to (s \cdot t){:}G)$ in $B(S)$.
- If $+ \in S$, include the axiom schemes $s{:}F \to (s+t){:}F$ and $t{:}F \to (s+t){:}F$ in $B(S)$.
- If $! \in S$, include the axiom scheme $t{:}F \to !t{:}t{:}F$ in $B(S)$.

All justification logics share the rule *modus ponens* (from $F \to G$ and F conclude G), but they differ in their treatment of constants. Quoting Fitting, "Constant symbols are intended to serve as justification of truths we cannot further analyze, but our ability to analyze is dependent on available machinery." Thus, variations in the rules governing constants.

- *The Axiom Necessitation Rule*: If A is an axiom an c is a constant, then $c{:}A$ is a theorem.
- *The Iterated Axiom Necessitation Rule*: If A is an axiom and $c_1, c_2, \ldots, c_n$ are constants, then $c_1{:}c_2{:}\cdots{:}c_n{:}A$ is a theorem.
- *The Theorem Necessitation Rule*: If X is a theorem and c is a constant, then $c{:}X$ is a theorem.

If both $\cdot$ and ! are in S, then $B(S)$ needs only the Axiom Necessitation Rule. If S has $\cdot$ but lacks !, then $B(S)$ requires the Iterated Axiom Necessitation Rule, and if S lacks $\cdot$, then $B(S)$ requires the Theorem Necessitation Rule.

In what was just described above, each constant may serve as justification for any axiom. One may also restrict the roles of various constants by means of a *constant specification*, associating individual constants with sets of individual instances of axioms. (In the case of the Iterated Axiom Necessitation rule, finite sequences of constants are associated with sets of axiom instances, and in the case of the Theorem Necessitation Rule constants are associated with sets of theorems, of course.) When each constant is associated with all axioms, the constant specification is called *full*.

Since $\{\cdot, +, !\}$ has eight subsets, we have just defined eight logics of justified belief, but as far as I know, those lacking $\cdot$ have never been studied or found any application. $B(\{\cdot, +\})$ is also known as **J**, the basic logic of justification and $B\{\cdot, +, !\}$ is known as **J4**; both were defined by Brezhnev ([6]). Their counterparts without $+$ are known as $\mathbf{J}^-$ and $\mathbf{J4}^-$ and were defined by Fitting in [7].

2.3 A Simple Semantics

The original intended semantics for **LP** was arithmetic proofs, but a more adaptable semantics was defined by Fitting in [7], generalizing the idea of Kripke models for modal logics. We will not have need of the full strength of Fitting models here, though, and will revert to an older semantics due to Mkrtychev [8]. Mkrtychev models are essentially one-world Fitting models.

The first notion we will need is that of an *evidence function*, which is simply any function $\mathcal{E}$ from justification terms in $L(S)$ to sets of $L(S)$ formulas. We may impose additional conditions on $\mathcal{E}$:

- If $\cdot$ is in S, we will insist that whenever $F \rightarrow G \in \mathcal{E}(s)$ and $F \in \mathcal{E}(t)$ it is also the case that $G \in \mathcal{E}(s \cdot t)$.
- If $+$ is in S, we will insist that $\mathcal{E}(s) \cup \mathcal{E}(t) \subseteq \mathcal{E}(s + t)$.
- If ! is in S, we will insist that whenever $F \in \mathcal{E}(t)$, it is also the case that $t{:}F \in \mathcal{E}(!t)$.

Finally, for an evidence function to be appropriate for a language $L(S)$ and a particular constant specification $\mathcal{C}$ in that language, it must behave properly on constants.

- If $\cdot$ and ! are both in S, it must be that $A \in \mathcal{E}(c)$ for each axiom $A \in \mathcal{C}(c)$ where c is a justification constant.
- If $\cdot$ is in S but ! is not in S, it must be that $c_2 : \cdots : c_n : A \in \mathcal{E}(c_1)$, $c_3{:}\cdots{:}c_n{:}A \in \mathcal{E}(c_2)$, $\ldots c_n{:}A \in \mathcal{E}(c_{n-1})$, and $A \in \mathcal{E}(c_n)$ for each axiom $A \in \mathcal{C}(\langle c_1, c_2 \ldots, c_n\rangle)$ where c_1, c_2, $\ldots$, c_n are justification constants. ($n \geq 1$.)
- If S lacks $\cdot$, it must be that $F \in \mathcal{E}(c)$ for each theorem $F \in \mathcal{C}(c)$ where c is a justification constant.

A *(Mkrtychev) structure* $\mathcal{M}$ for a language $L(S)$ is a pair $\langle \mathcal{E}, \mathcal{V}\rangle$ where $\mathcal{E}$ is an evidence function appropriate to $L(S)$ and $\mathcal{V}$ is a propositional valuation.

We will define satisfaction of a formula F in a structure $\mathcal{M}$ (written $\mathcal{M} \Vdash F$) as follows:

- $\mathcal{M} \Vdash P$ for propositional variable P if and only if $\mathcal{V}(P)$ is true.
- $\mathcal{M} \nVdash \bot$.
- $\mathcal{M} \Vdash F \rightarrow G$ if and only if either $\mathcal{M} \nVdash F$ or $\mathcal{M} \Vdash G$.
- $\mathcal{M} \Vdash t{:}F$ if and only if $F \in \mathcal{E}(t)$.

Kuznets ([9]) proved the soundness and completeness of Mkrtychev models for $B(\{\cdot, +\})$ and $B(\{\cdot, +, !\})$. Essentially the same proof goes through for $\cdot$-free and $+$-free logics of justified belief.

3 Conservativity

While Fitting's approach to logics of knowledge [1] was entirely syntactic, my approach will be entirely semantic, for reasons mentioned in the concluding

section of this note. I will show that Mkrtychev models of weaker logics can be extended to models of stronger logics while preserving the truth of formulas from the less-expressive language. I will focus on extending $B(\{\cdot,+\})$ to $B(\{\cdot,+,!\})$ (that is, **J** to **J4**) but essentially the same argument works when adding + or even $\cdot$ to a language.

Before I state the main result, a note constant specifications. While Iterated Constant Necessitation is not necessary in logics with positive introspection (!), it does no harm to include it. So if we want to find a constant specification for **J4** which will be conservative over **J** with some constant specification, the **J4**-constant specification will need to extend the **J**-constant specification (which might have permitted, for example, $c_1 : c_2 : c_3 : (P \to P \to P)$). In our first result, we will not extend the constant specification, but simply leave it intact.

Theorem 1. *Let $\mathcal{C}$ be a constant specification in $L(\{\cdot,+\})$. If $F \in L(\{\cdot,+\})$ is provable in $B(\{\cdot,+,!\})$ (that is, **J4**) under constant specification $\mathcal{C}$, then F is provable in $B(\{\cdot,+\})$ (that is, **J**) under constant specification $\mathcal{C}$.*

Proof. We will establish that any Mkrtychev model for **J** can be extended to a Mkrtychev model for **J4** which preserves the truth of !-free formulas. By the completeness of Mkrtychev models for **J**, if F is not provable in **J**, then there will be Mkrtychev model for **J** making F false. This will be extended to a **J4** model in which F is also false, showing that F was not provable in **J4** by the soundness of Mkrtychev models.

We begin with a Mkrtychev model $\mathcal{M}$ for **J**. Recall that this means that we have a propositional valuation $\mathcal{V}$ and an evidence function $\mathcal{E}$ with the properties that $\mathcal{E}(s) \cup \mathcal{E}(t) \subseteq \mathcal{E}(s+t)$ and whenever $F \to G \in \mathcal{E}(s)$ and $F \in \mathcal{E}(t)$ it is also the case that $G \in \mathcal{E}(s \cdot t)$. The handy thing about a semantic approach logics of belief is that beliefs need not have anything to do with the "real world" so we do not have to worry any further about our propositional valuation $\mathcal{V}$.

We will extend $\mathcal{E}$ to a **J4**-appropriate evidence function $\mathcal{E}'$ in stages. Because we are leaving the constant specification alone, we need only one additional property: that whenever $F \in \mathcal{E}'(t)$ we also have $t : F \in \mathcal{E}'(!t)$. (We essentially taking the transitive closure of $\mathcal{E}$.)

We will define $\mathcal{E}_n$ recursively, treating constants at the initial stage and closure under operations at successive stages. We can then set $\mathcal{E}' = \bigcup_{n=0}^{\infty} \mathcal{E}_n$.

We begin by setting $\mathcal{E}_0 = \mathcal{E}$. Now we can define $\mathcal{E}_{n+1}$.

- If c is a justification constant, $\mathcal{E}_{n+1}(c) = \mathcal{E}_n(c)$.
- If x is a justification variable, $\mathcal{E}_{n+1}(x) = \mathcal{E}_n(x)$.
- $\mathcal{E}_{n+1}(s+t) = \mathcal{E}_n(s+t) \cup \mathcal{E}_n(s) \cup \mathcal{E}_n(t)$.
- $\mathcal{E}_{n+1}(s \cdot t) = \mathcal{E}_n(s \cdot t) \cup \{G | F \to G \in \mathcal{E}_n(s) \text{ and } F \in \mathcal{E}_n(t)\}$.
- $\mathcal{E}_{n+1}(!t) = \mathcal{E}_n(!t) \cup \{t : F | F \in \mathcal{E}_n(t)\}$.

The only evidence for formulas containing the ! operator will be justification terms which themselves contain !.

Lemma 1. *If t is !-free and $F \in \mathcal{E}_n(t)$, then F is !-free as well.*

Proof. We will prove this by induction, and the base ($n = 0$) case is trivial, since $\mathcal{E}_0(t) = \mathcal{E}(t)$ and $\mathcal{E}$ was an evidence function in a !-free language.

For the inductive step of the proof, let us assume that $F \in \mathcal{E}_{k+1}(t)$. If $F \in \mathcal{E}_k(t)$ as well, we are finished by our inductive hypothesis. So let us assume that $F \notin \mathcal{E}_k(t)$. It is impossible by the definition of $\mathcal{E}_{k+1}$ that t is a justification constant or justification variable, so t is either $u \cdot v$ or $u + v$. (Recall that t was !-free.)

- If $t = u \cdot v$, then there is some G with $G \to F \in \mathcal{E}_k(u)$ and $G \in \mathcal{E}_k(v)$. By our inductive hypothesis, $G \to F$ is !-free, so F will be as well.
- If $t = u + v$, then either $F \in \mathcal{E}_k(u)$ or $F \in \mathcal{E}_k(v)$. In either case, we know by our inductive hypothesis that F is !-free.

This ends the inductive argument and the proof of the first lemma.

Now we can get almost all of the way home with a second lemma.

Lemma 2. *Let t and F be !-free. $F \in \mathcal{E}'(t)$ if and only if $F \in \mathcal{E}(t)$.*

Proof. That $F \in \mathcal{E}(t)$ implies $F \in \mathcal{E}'(t)$ is immediate from the construction of $\mathcal{E}'$.

To show the converse, we will prove that if $F \in \mathcal{E}_n(t)$ then $F \in \mathcal{E}(t)$ by induction on n. Again, the base case is trivial.

For the induction, we may assume that if $G \in \mathcal{E}_k(s)$ then $G \in \mathcal{E}(s)$ for all !-free pairs s and G. We wish to show that if F and t are !-free and $F \in \mathcal{E}_{k+1}(t)$ then $F \in \mathcal{E}(t)$. If $F \in \mathcal{E}_k(t)$, we are done immediately by our inductive hypothesis. So let us examine the other possible cases:

- If $t = u \cdot v$ and there is $G \in \mathcal{E}_k(v)$ with $G \to F \in \mathcal{E}_k(u)$. Because $u \cdot v$ is !-free, we know by our earlier lemma that it is also the case that $G \to F$ and G are !-free. By our inductive hypothesis, $G \to F \in \mathcal{E}(v)$ an $G \in \mathcal{E}(u)$. Since $\mathcal{E}$ was an evidence function, it must be that $F \in \mathcal{E}(u \cdot v)$.
- If $t = u + v$ and $F \in \mathcal{E}_k(u)$, then by our inductive hypothesis, $F \in \mathcal{E}(u)$. Because $\mathcal{E}$ was an evidence function, we know that $\mathcal{E}(u) \subseteq \mathcal{E}(u + v)$, so $F \in \mathcal{E}(u + v)$. The case for $F \in \mathcal{E}_k(v)$ is identical.

This completes the induction and the proof of our second lemma.

It is clear from the definition of $\mathcal{E}'$ that it is an evidence function appropriate to the logic **J4**, so we can define a **J4** Mkrtychev model $\mathcal{M}' = \langle \mathcal{E}', \mathcal{V} \rangle$ where $\mathcal{V}$ is the propositional valuation from our original **J**-model $\mathcal{M}$. What remains to be shown is that if F is a !-free formula, then $\mathcal{M} \Vdash F$ if and only if $\mathcal{M}' \Vdash F$. We can prove this by a very easy induction on the construction of F.

Because both $\mathcal{M}$ and $\mathcal{M}'$ are built from the same propositional valuation $\mathcal{V}$ the case for propositional variables is immediate, as is the case for $\bot$. That $\mathcal{M} \Vdash t : G$ if and only if $\mathcal{M}' \Vdash t : G$ is immediate from the definition of $\Vdash$ and the second lemma above. The argument for $F = G_1 \to G_2$ is standard and straightforward.

Of course we do not want to hamstring ourselves with constant specifications for **J4** which are entirely !-free. In particular, it would be good to have conservativity hold for axiomatically appropriate constant specifications. With a few reasonable conditions, we can generalize Theorem 1 to a broader class of constant specifications.

In the next theorem, I will assume that my constant specification for **J** is *schematic*. In the context of Iterated Constant Necessitation, that means that if a particular axiom A is in $\mathcal{C}(\langle c_1, c_2, \ldots, c_n\rangle)$ for some sequence c_1, c_2, $\ldots$, c_n of justification constants, then so are all other instances of the schema of which A is an instance. In other words, entire schemas are specified by a particular sequence of justification constants.

Secondly, I will assume that any constants in **J4** used to justify instances of positive introspection ($t{:}F \rightarrow !t{:}t{:}F$) will be new constants not appearing in the original constant specification for **J**. (New axioms, new constants.) This would be a corollary of having a schematically injective constant specification.[1] For the sake of simplicity, I will assume that there is a single such new constant and refer to it as $c_!$.

Let $\mathcal{C}_1$ be a schematic constant specification for **J**, and let $\mathcal{C}_1(c_!) = \emptyset$. Let $\mathcal{C}_2$ extend $\mathcal{C}_1$ to a constant specification for **J4** by the following:

- If $A \in \mathcal{C}_1(c)$ for an axiom A in $L(\{\cdot, +\})$ and A' is an instance of that same axiom scheme but in the extended language $L(\{\cdot, +, !\})$, put A' into $\mathcal{C}_2(c)$.
- If A is a instance of the positive introspection scheme $t{:}F \rightarrow !t{:}t{:}F$, put A into $\mathcal{C}_2(c_!)$.

Note that if $\mathcal{C}_1$ is axiomatically appropriate, so will $\mathcal{C}_2$ be, though with unnecessary instances of iterated justification constants. If we start with any schematic constant specification $\mathcal{C}$ for **J4** with a single justification constant $c_!$ whose sole role is as evidence of the positive introspection scheme, we can find a constant specification $\mathcal{C}_1$ for **J** which extends to $\mathcal{C}$.

Theorem 2. *Let constant specification $\mathcal{C}_2$ extend constant specification $\mathcal{C}_1$ as defined above. If $F \in L(\{\cdot, +\})$ and containing no instances of $c_!$ is provable in $B(\{\cdot, +, !\})$ (that is, **J4**) under constant specification $\mathcal{C}_2$, then F is provable in $B(\{\cdot, +\})$ (that is, **J**) under constant specification $\mathcal{C}_1$.*

Proof. We will again establish that any Mkrtychev model for **J** can be extended to a Mkrtychev model for **J4** which preserves the truth of !-free and $c_!$-free formulas, subject to the conditions on constants mentioned just above. The proof will be nearly identical in its outline to that of Theorem 1, but there will be one small change to the construction of the extension $\mathcal{E}'$ of the evidence relation $\mathcal{E}$, entailing some extra work in the lemmas.

As above, we begin with a Mkrtychev model $\mathcal{M}$ for **J** which respects the constant specification $\mathcal{C}_1$, consisting of a propositional valuation $\mathcal{V}$ and an evidence function $\mathcal{E}$.

[1] In a *schematically injective* constant specification, each constant corresponds to either no axioms at all or all instances of a single axiom schema.

This time when we extend $\mathcal{E}$ to a **J4**-appropriate evidence function $\mathcal{E}'$ we will need two additional properties: First, we need to extend the behavior of constants in $\mathcal{E}$ to include axioms from $L(\{\cdot, +, !\})$, and as before we need it to be the case that whenever $F \in \mathcal{E}'(t)$ we also have $t{:}F \in \mathcal{E}'(!t)$.

We will again define $\mathcal{E}_n$ recursively, treating constants at the initial stage and closure under operations at successive stages. We can still set $\mathcal{E}' = \bigcup_{n=0}^{\infty} \mathcal{E}_n$.

We begin with $\mathcal{E}_0$.

- If t is not a justification constant, let $\mathcal{E}_0(t) = \mathcal{E}(t)$.
- If c is a justification constant, let $\mathcal{E}_0(c) = \mathcal{E}(c) \cup \mathcal{C}_2(c)$

Now we can define $\mathcal{E}_{n+1}$ exactly as in Theorem 1.

- If c is a justification constant, $\mathcal{E}_{n+1}(c) = \mathcal{E}_n(c)$.
- If x is a justification variable, $\mathcal{E}_{n+1}(x) = \mathcal{E}_n(x)$.
- $\mathcal{E}_{n+1}(s + t) = \mathcal{E}_n(s + t) \cup \mathcal{E}_n(s) \cup \mathcal{E}_n(t)$.
- $\mathcal{E}_{n+1}(s \cdot t) = \mathcal{E}_n(s \cdot t) \cup \{G | F \to G \in \mathcal{E}_n(s) \text{ and } F \in \mathcal{E}_n(t)\}$.
- $\mathcal{E}_{n+1}(!t) = \mathcal{E}_n(!t) \cup \{t{:}F | F \in \mathcal{E}_n(t)\}$.

At this point, we will need some notation and somewhat more complicated version of Lemma 1.

Notation: If F is a formula in $L(\{\cdot, +, !\})$, let $(F)^{\ddagger}$ be F with all instances of any justification term $!t$ replaced with the fresh justification variable y, and all instances of the justification constant $c_!$ replaced with c_0 where $P \to (P \to P) \in \mathcal{C}_1(c_0)$. Of course if F contains no occurrences of either ! or $c_!$, then $(F)^{\ddagger} = F$.

Revised Lemma 1. *If t is !- and $c_!$-free, then if $F \in \mathcal{E}_n(t)$, we have $(F)^{\ddagger} \in \mathcal{E}_n(t)$ as well.*

Proof. We will prove this, of course, by induction. First, for $n = 0$. If t is not a justification constant and $F \in \mathcal{E}_0(t)$, then $F \in \mathcal{E}(t)$. Because $\mathcal{E}$ was the evidence function for a **J**-model in a language without ! or $c_!$, we know that F is !- and $c_!$-free, meaning that $(F)^{\ddagger}$ is identical to F. A similar argument works in the case that t is a justification constant and $F \in \mathcal{E}(c)$.

To complete the base case of the induction, we need to show that if $F \in \mathcal{C}_2(c)$, $(F)^{\ddagger} \in \mathcal{C}_2(c)$ as well. That is, we need to show that if F is an instance of a **J4** axiom schema other than positive introspection (recall that $c \neq c_!$) then $(F)^{\ddagger}$ is also an instance of that same schema. However, this is immediate from the fact that $\ddagger$ leaves intact all instances of the justification operators $\cdot$ and $+$ and all propositional connectives. (For example, $(s + t{:}G)^{\ddagger} = (s)^{\ddagger} + (t)^{\ddagger}{:}(G)^{\ddagger}$.)

For the inductive step of the proof, let us assume that $F \in \mathcal{E}_{k+1}(t)$. If $F \in \mathcal{E}_k(t)$ as well, we are finished by our inductive hypothesis. So let us assume that $F \notin \mathcal{E}_k(t)$. It is impossible by the definition of $\mathcal{E}_{k+1}$ that t is a justification constant or justification variable, so t is either $u \cdot v$ or $u + v$. (Recall that t was !- and $c_!$-free.)

- If $t = u \cdot v$, then there is some G with $G \to F \in \mathcal{E}_k(u)$ and $G \in \mathcal{E}_k(v)$. By our inductive hypothesis, $(G \to F)^\ddagger \in \mathcal{E}_k(u)$ and $(G)^\ddagger \in \mathcal{E}_k(v)$. Since $\ddagger$ is not concerned with propositional connectives, $(G \to F)^\ddagger = (G)^\ddagger \to (F)^\ddagger$. Since we have $(G)^\ddagger \in \mathcal{E}_k(v)$ and $(G)^\ddagger \to (F)^\ddagger \in \mathcal{E}_k(u)$, the definition of $\mathcal{E}_{k+1}$ tells us that $(F)^\ddagger \in \mathcal{E}_{k+1}(u \cdot v)$.
- If $t = u+v$, then either $F \in \mathcal{E}_k(u)$ or $F \in \mathcal{E}_k(v)$. By our inductive hypothesis, $(F)^\ddagger \in \mathcal{E}_k(u)$ or $(F)^\ddagger \in \mathcal{E}_k(v)$. By the definition of $\mathcal{E}_{k+1}$, $(F)^\ddagger \in \mathcal{E}_{k+1}(u+v)$.

This ends the inductive argument and the proof of the first lemma.

The complexities of the first lemma lead to a few changes in the proof of second lemma as well.

Revised Lemma 2. *Let t and F be !- and $c_!$-free. $F \in \mathcal{E}'(t)$ if and only if $F \in \mathcal{E}(t)$.*

Proof. That $F \in \mathcal{E}(t)$ implies $F \in \mathcal{E}'(t)$ is again immediate from the construction of $\mathcal{E}'$.

To show the converse, we will as usual prove that if $F \in \mathcal{E}_n(t)$ then $F \in \mathcal{E}(t)$ by induction on n.

If t is not a justification constant, then $\mathcal{E}_0(t) = \mathcal{E}(t)$. If t is a justification constant $c \neq c_!$, $\mathcal{C}_1$ and $\mathcal{C}_2$ agree on F by the definition of $\mathcal{C}_2$. (Recall that F was !- and $c_!$-free.) Thus, if $F \in \mathcal{E}_0(t)$ then $F \in \mathcal{E}(t)$.

Now we may assume that if $G \in \mathcal{E}_k(s)$ then $G \in \mathcal{E}(s)$ for all !- and $c_!$-free pairs s and G. We wish to show that if F and t are !- and $c_!$-free and $F \in \mathcal{E}_{k+1}(t)$ then $F \in \mathcal{E}(t)$. If $F \in \mathcal{E}_k(t)$, we are done immediately by our inductive hypothesis. So let us examine the other possible cases:

- If $t = u \cdot v$ and there is $G \in \mathcal{E}_k(v)$ with $G \to F \in \mathcal{E}_k(u)$. Because $u \cdot v$ is !- and $c_!$-free, we know by our earlier lemma that it is also the case that $(G)^\ddagger \in \mathcal{E}_k(v)$ and $(G \to F)^\ddagger \in \mathcal{E}_k(u)$. Because we can move $\ddagger$ past propositional connectives, and because F is !- and $c_!$-free, we know that $(G \to F)^\ddagger = (G)^\ddagger_\ddagger \to (F)^\ddagger = (G)^\ddagger \to F$. Thus, we have $(G)^\ddagger \in \mathcal{E}_k(v)$ and $(G)^\ddagger \to F \in \mathcal{E}_k(u)$. By our inductive hypothesis, $(G)^\ddagger \in \mathcal{E}(v)$ and $(G)^\ddagger \to F \in \mathcal{E}(u)$. Because $\mathcal{E}$ was an evidence function for a Mkrtychev model for **J**, it must be the case that $F \in \mathcal{E}(u \cdot v)$.
- If $t = u + v$ and $F \in \mathcal{E}_k(u)$, then by our inductive hypothesis, $F \in \mathcal{E}(u)$. Because $\mathcal{E}$ was an evidence function, we know that $\mathcal{E}(u) \subseteq \mathcal{E}(u + v)$, so $F \in \mathcal{E}(u + v)$. The case for $F \in \mathcal{E}_k(v)$ is identical. (This case is unchanged from the original version of the lemma.)

This completes the induction and the proof of our second lemma.

The remainder of the proof of the current theorem is both standard and identical with the end of the proof of Theorem 1.

Similar proofs work to show the conservativity of, say, **J4** over $\mathbf{J4}^-$ (the +-free fragment of **J4**). (In fact, the proofs of conservativity over +-free fragments

have much less need for equivocation about constant specifications.) The only substantial differences would come in the inductive steps of the lemmas.

For example, in the (revised) first lemma, we would need the following argument (for **J4** over $\mathbf{J4}^-$, assuming that $(F)^-$ was defined analogously with $(F)^\ddagger$):

- If $t = !s$, then $F = s{:}G$ and $G \in \mathcal{E}_k(s)$. By our inductive hypothesis[2] we know that $(G)^- \in \mathcal{E}_k(s)$ as well. Because t was $+$-free, so is s, so $(s{:}G)^- = s{:}(G)^-$. Because $(G)^- \in \mathcal{E}_k(s)$, $s{:}(G)^- \in \mathcal{E}_{k+1}(!s)$, and so $(s{:}G)^- \in \mathcal{E}_{k+1}(!s)$.

And in the second:

- If $t = !s$ and $F = s : G$ for some $G \in \mathcal{E}_k(s)$, then we know by our inductive hypothesis[3] that $G \in \mathcal{E}(s)$. Because $\mathcal{E}$ was a $\mathbf{J4}^-$-appropriate evidence function, it must be that $s{:}G \in \mathcal{E}(!s)$.

The interested reader can work out details for other cases.

4 Consistent Belief

What we have been examining so far could be called the logic of "pure normal belief" (with or without positive introspection). From a modal standpoint, the only axioms are $\Box(F \to G) \to \Box F \to \Box G$ and possibly $\Box F \to \Box\Box F$. No other constraints are placed on what is believed. In logics of knowledge, consistency of belief is automatic because things believed/known are also true, and no inconsistency can be true. However, one can fairly simply mandate consistency of beliefs without requiring that all which is believed be true. The modal axiom **D** (from the word deontic[4]) $((\Box\bot) \to \bot)$ accomplishes this. This can also be introduced in logics of justified belief as the axiom scheme $(t{:}\bot) \to \bot$.

In a way, insisting on consistent belief can be seen as a middle ground between unconstrained (normal) belief and knowledge. This makes it surprising that while **J4** is conservative over **J** and **JT4** over **JT**, it is not the case that **JD4** is conservative over **JD**. In other words, introspection can introduce inconsistencies in otherwise consistent belief systems.

The potential that introspection has for havoc can be seen from a simple example. I might both believe that the sky is blue and believe that I do not believe that the sky is blue. (This could be expressed as $x : P$ and $y : ((x : P) \to \bot)$.) Absent positive introspection, I can hold both these beliefs. But in the presence of positive introspection (and application), an inconsistent belief appears. (From $x{:}P$ deduce $!x{:}x{:}P$, and by application $(y{\cdot}!x){:}\bot$.)

The same example works in the absence of the operator $+$. I have not explored the conservativity of **JD** over $\mathbf{JD}^-$ or of **JD4** over $\mathbf{JD4}^-$.

[2] Having, of course, to do with $+$-free justification terms and formulas $(G)^-$ which have had the $+$-terms stripped out.

[3] Again, this would be a different hypothesis than in our original version of the lemma.

[4] The word *deontic* denotes a connection to duty or obligation. The more common form of the axiom **D** is $\Box F \to \Diamond F$, which could be interpreted as "What is mandatory is also permitted." In normal modal logics, this scheme is equivalent to our formulation.

5 Conclusion

In Fitting's proof of conservativity for logics of knowledge [1], he showed that not only conclusions but entire proofs could be preserved by careful elimination of justification operators. However, his reduction relied heavily on the presence of the truth axiom ($t\!:\!F \to F$) and its variants. For example, if we were trying to eliminate occurrences of ! from a proof, and the justification term t contained ! (while u did not), the axiom $u : F \to (t + u) : F$ would become $u : F \to F$, an instance of the truth axiom.

Because the truth axiom is absent from logics of belief, Fitting's approach does not immediately generalize. What makes syntactic approaches to this problem so difficult (at least for logics with the $\cdot$ operator on justification terms) is that, for example, the operator ! can show up in wholly inessential ways in an internally cut formula F while not appearing at all in the justifications for F. (By an "internally cut" formula, I mean the F from the axiom $s\!:\!(F \to G) \to (s\!:\!F \to (s \cdot t)\!:\!G)$. F might contain instances of ! introduced in irrelevant ways.) This same difficulty presents itself, as far as I could tell, regardless of whether one is dealing with Hilbert-style proofs or sequent/tableau proofs.

Thus, one problem clearly still open in this area is the existence or impossibility of direct translations of proofs in a stronger logic of belief to those in a weaker logic of belief.

Also still open is conservativity of logics of belief with the negative introspection (?) operator. Because the arguments in the present paper relied heavily on the monotonicity of the construction of the extension of evidence relations, they would seem incompatible with negative introspection.

References

1. Fitting, M.: Explicit logics of knowledge and conservativity. In: Tenth International Symposium on Artificial Intelligence and Mathematics, ISAIM 2008, Fort Lauderdale, Florida, January 2–4, 2008, Online Proceedings (2008)
2. Artëmov, S.N.: Logic of proofs. Annals of Pure and Applied Logic 67(1–3), 29–59 (1994)
3. Artemov, S.N.: Explicit provability: the intended semantics for intuitionistic and modal logic. Technical Report CFIS 98–10, Cornell University (September 1998)
4. Artemov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic 7(1), 1–36 (2001)
5. Gödel, K.: Eine interpretation des intuitionistischen aussagenkalküls. Ergebnisse Math. Colloq. 4, 39–40 (1933)
6. Brezhnev, V.N.: On the logic of proofs. In: Striegnitz, K. (ed.) Proceedings of the sixth ESSLLI Student Session, Helsinki, pp. 35–46 (August 2001)
7. Fitting, M.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132(1), 1–25 (2005)
8. Mkrtychev, A.: Models for the Logic of Proofs. In: Adian, S.I., Nerode, A. (eds.) LFCS 1997. LNCS, vol. 1234, pp. 266–275. Springer, Heidelberg (1997)
9. Kuznets, R.: On the complexity of explicit modal logics. In: Clote, P.G., Schwichtenberg, H. (eds.) CSL 2000. LNCS, vol. 1862, pp. 371–383. Springer, Heidelberg (2000)

Unifying Sets and Programs via Dependent Types

Wojciech Moczydłowski

Google, New York
wojtekm@google.com

Abstract. We present a foundational framework, which we call D, unifying a lazy programming language with an impredicative constructive set theory IZF_R by means of dependent types. We show that unification brings many benefits to both worlds. First, D supports two paramount paradigms of creating reliable software: correctness by construction and post-construction verification, while retaining the expressiveness of set theory. Second, D provides new expressive power, which makes it possible to internalize and prove *inside* D the standard meta-theoretic properties of constructive systems, such as Numerical Existence Property and Program Extraction. Finally, computation arising from the programming language significantly enriches set theory, as we show that D is stronger than IZF_R and that its real numbers behave in a better way.

1 Introduction

Consider a simple recursive program:

```
f(0) = 0
f(n+1) = f(n) + 1
```

It is straightforward to prove that $\forall n \in \omega.\ f(n) = n$. Just proceed by induction on n, and the claim follows.

But is it really that simple? Just where exactly does this argument take place, and how easy is it to fully formalize it? Two major answers are:

1. The argument is done where all mathematical developments are done: in set theory. Numbers are formalized in the standard way and recursion is defined using the Recursion Theorem. The syntax of the programming language can be formalized using for example Gödel numbering or hereditarily finite sets, and there are plenty of semantics to choose from.
2. The argument is done in a logic designed from the start to reason about programs, embedding programs and computation deeply inside, with an existing computer tool which can be used for formalization. Prominent examples of such logics are Higher Order Logic [1], versions of type theory [2,3] and MinLog [4].

Sadly, none of the answers is really satisfactory. The formalization of arguments in set theory, although a standard procedure from the mathematical point of

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 365–379, 2009.

view, is very difficult to apply in practice. The sheer amount of formalizations needed to prove the recursion theorem and difficulties associated with formalization of the syntax are probably the reason why to this day there exists only one prover based on set theory used in practice to reason about programs [5]. Due to its closed nature, its capabilities are unclear.

These difficulties can be removed by using a logic designed with programs in mind, such as HOL, type theory or MinLog. The price to pay, however, is expressiveness and easiness of use. HOL is a very weak logic. The strongest modern applied type theories are much weaker than ZFC, the standard foundation of mathematics. It also remains to be seen whether types can repeat the remarkable success of sets as a foundational basis and tool for abstraction.

Our framework D provides a solution to these problems along with extra benefits. Briefly, D unifies a lazy programming language P with impredicative constructive set theory IZF with Replacement (IZF_R), using weakly dependent logic. Since P is an integral part of the logic, no time is lost on formalizing syntax and semantics. As the framework unifies sets and programs, the logic available for reasoning about the programs is the standard set theory. In this way D supports the post-construction verification paradigm — it is possible to write the program first and then use set theory to reason about it.

Furthermore, D possesses all properties desirable from the proof-theoretic point of view, including Subject Reduction, Progress and Normalization. Therefore, as we showed in the previous work [6,7], it also supports the correct-by-construction paradigm: programs can be extracted from set theoretic proofs. Moreover, D makes it possible to state and prove the properties of program extraction directly *inside* of D, instead of using convoluted metatheoretical constructions.

Finally, the combination significantly influences the set-theoretical side. We show that D is stronger than IZF_R, by showing that a countable version of the Axiom of Choice is derivable in D. The result implies that reals behave in D in a much better way than in IZF_R.

This paper is organized as follows. We present D in Sections 2 and 3 and develop mathematics in it in Section 4. Therein we show how sets influence programs and programs influence sets. Related work is discussed in Section 5.

2 The Informal Account of D

In this section we present our framework D informally. The fully formal account follows in Section 3. D is based on two pillars: a dependent variant of constructive set theory IZF_R, which we call IZF_T, and a lazy, functional programming language, which we call P. The framework unifies these two worlds together. As we shall see in Section 4, the unification makes it possible to use set theory to reason about programs of P and for the computation in P to influence set theory.

- (IN) $\forall a, b.\ a \in b \leftrightarrow \exists c.\ c \in_I b \wedge a = c$
- (EQ) $\forall a, b.\ a = b \leftrightarrow \forall d.\ (d \in_I a \rightarrow d \in b) \wedge (d \in_I b \rightarrow d \in a)$
- (EMPTY) $\forall c.\ c \in_I \emptyset \leftrightarrow \bot$
- (PAIR) $\forall a, b \forall c.\ c \in_I \{a, b\} \leftrightarrow c = a \vee c = b$
- (OMEGA) $\forall c.\ c \in_I \omega \leftrightarrow c = \emptyset \vee \exists b \in \omega.\ c = b \cup \{b, b\}$
- ($\mathrm{SEP}_{\phi(p,a,\boldsymbol{f})}$) $\forall \boldsymbol{f}, a \forall c.\ c \in_I S_{\phi(p,a,\boldsymbol{f})}(a, \boldsymbol{f}) \leftrightarrow (p : c \in a) \wedge \phi(p, c, \boldsymbol{f})$
- (UNION) $\forall a \forall c.\ c \in_I \bigcup a \leftrightarrow \exists b \in a.\ c \in b$
- (POWER) $\forall a \forall c.\ c \in_I P(a) \leftrightarrow \forall b.\ b \in c \rightarrow b \in a$
- ($\mathrm{REPL}_{\phi(p,a,b,\boldsymbol{f})}$) $\forall \boldsymbol{f}, a \forall c.\ c \in_I R_{\phi(p,a,b,\boldsymbol{f})}(a, \boldsymbol{f}) \leftrightarrow (\forall x.\ (p : x \in a) \rightarrow \exists! y.\ \phi(p, x, y, \boldsymbol{f})) \wedge (\exists x.\ (p : x \in a) \wedge \phi(p, x, c, \boldsymbol{f}))$
- ($\mathrm{IND}_{\phi(a,\boldsymbol{f})}$) $\forall \boldsymbol{f}.\ (\forall a.\ (\forall b.\ b \in_I a \rightarrow \phi(b, \boldsymbol{f})) \rightarrow \phi(a, \boldsymbol{f})) \rightarrow \forall a.\ \phi(a, \boldsymbol{f})$

Fig. 1. The axioms of IZF_T

$$P ::= x \mid \lambda x.\ P \mid 0 \mid S(P) \mid \mathrm{casenat}(P, Q, x.R) \qquad [\circ] ::= [\circ]\ \ P \mid \mathrm{casenat}([\circ], Q, x.R)$$

$$(\lambda x.\ P)\ O \rightarrow O[x := P] \qquad \mathrm{casenat}(0, Q, x.R) \rightarrow Q$$
$$\mathrm{casenat}(S(P), Q, x.R) \rightarrow R[x := \mathrm{casenat}(P, Q, x.R)]$$

$$\frac{}{\Gamma \vdash 0 : \mathrm{nat}} \qquad \frac{\Gamma \vdash P : \mathrm{nat}}{\Gamma \vdash S(P) : \mathrm{nat}} \qquad \frac{\Gamma \vdash P : \mathrm{nat} \quad \Gamma \vdash Q : \phi \quad \Gamma, x : \phi \vdash R : \phi}{\Gamma \vdash \mathrm{casenat}(P, Q, x.R) : \phi}$$

Fig. 2. The programming language P

The theory IZF_R, first introduced by Myhill [8], is a constructive counterpart of ZF set theory in its version with Replacement. IZF_T is a dependent extension of IZF_R. The underlying logic of IZF_T is the constructive first-order logic extended with dependent conjunctions, dependent implications and the type of natural numbers.

The axioms of IZF_T are the same as in [9]: Empty Set, Pairing, Infinity, Union, Power Set, $\in$-Induction, dependent Separation and dependent Replacement. They are listed in Figure 1. As usual [9,7], an intensional membership relation $\in_I$ is used as a building block for Extensionality and Leibniz axioms.

The programming language P is a lazy lambda calculus, with natural numbers and the recursion combinator. We summarize it in Figure 2. Our framework does not depend heavily on the choice of P; any reasonable functional programming language with type-theoretic semantics could be used instead.

Just as the set-theoretical layer of D makes it possible to reason about equality of sets, its programming part makes it possible to reason about computational equivalence of programs. We define the relation of computational equivalence as the smallest contextually closed equivalence relation on programs containing the reduction relation. We write $P \equiv Q$, if P is computationally equivalent to Q. For example, $0 \equiv 0, (\lambda x.\ x)\ 0 \equiv 0$ and $\lambda y.\ (\lambda x.\ x)\ 0 \equiv \lambda y.\ 0$.

The most difficult part of D to state informally is the glue between the world of programming languages and the world of sets. If the reader feels that our presentation is too informal, we recommend skipping to Section 3. We adopt the notation $M : \phi$ for the fact that M is a proof of ϕ. As D is a

dependent theory, these judgments are as integral to D as set-theoretical membership formulas $A \in B$. We can read $M : \phi$ as "M is a proof of ϕ" or "M proves ϕ".

First, for any program P of type nat, there is a corresponding element of ω, which we denote by $\overline{P}$. We define this injecting map so that $\overline{0} = \emptyset$, $\overline{S(P)} = \overline{P} \cup \{\overline{P}\}$ (recall that in set theory, $n \cup \{n\}$ denotes $n + 1$) and so that it is a homomorphism wrt. computational equivalence and set equality: if $P \equiv Q$, then $\overline{P} = \overline{Q}$.

Second, if $M : t \in \omega$, then there is a corresponding program $\mathrm{prog}(M)$ of type nat. This map is defined so that if $M : \emptyset \in \omega$, then $\mathrm{prog}(M) \equiv 0$ and if $M : t \cup \{t\} \in \omega$, then $\mathrm{prog}(M) \equiv S(\mathrm{prog}(N))$, where $N : t \in \omega$ results from M in a natural way. Finally, if M, N, O prove $t \in \omega, u \in \omega$ and $t = u$, respectively, then $\mathrm{prog}(M) \equiv \mathrm{prog}(N)$.

Although D might seem overwhelming at the first sight, we think these are necessary ingredients to make D a proof-theoretically solid setting, while at the same time a powerful programming language. Thanks to our axioms, a programmer can for example freely mix $\mathrm{prog}(M)$ terms with numbers entered by user, as they both exist on equal grounds. We hope an implementation of D would make an impact on bringing set theory closer to students, as it would enable them to play and program directly with set-theoretic objects, in addition to seeing them as static objects in textbooks and on blackboards. The reader might want to skim Section 4 to see how D is used in mathematics, before delving into formalities of the next section.

3 The Formal Account of D

In this section we provide more detailed presentation of D. While it is essentially self-contained, our previous work [7] provides ample extra background on the design of the system.

3.1 The Terms of *D*

The terms of D are divided into four syntactic categories, encompassing proof terms, programs, set terms and formulas. We will generally use letters M, N, O for proof terms, P, Q for programs, s, t, u for set terms, ϕ, ψ, ϑ for formulas and T, S for arbitrary terms. There are three kinds of variables. The first one, denoted by letters p, q, r, intuitively corresponds to the propositional implication. We call them *proof variables*. The second one, denoted by letters x, y, z, is used in the programming language part of D. Finally, the third one, usually denoted by letters a, b, c, intuitively corresponds to the first-order quantification. The notation $a, b.\ M$ stands for a term with its variables a, b bound. The notation $\boldsymbol{T}$ stands for a sequence of terms. The free variables of a term M are denoted by $FV(M)$.

The terms of D are defined by means of an abstract grammar. The first part of the grammar generates the proof terms. There are three groups of proof

terms. The first group of the proof terms corresponds to the first-order logic with dependent features:

$$M ::= p \mid M\ N \mid \lambda p.\ M \mid \mathrm{inl}(M) \mid \mathrm{inr}(M) \mid \mathrm{fst}(M) \mid \mathrm{snd}(M) \mid \langle M, N\rangle \mid \mathrm{magic}(M)$$
$$\mathrm{case}(M, p.\ N, p.\ O) \mid \lambda *\,.\ M \mid \mathrm{let}\ [*, p] := M\ \mathrm{in}\ N \mid [*, M] \mid M\ * \mid \mid \mathrm{in}(P) \mid \mathrm{ax}$$

Note that the first-order variables and set terms are not present in the proof terms. Instead, they are replaced by $*$, a new symbol of the language. This is because in set theory, the computational content of the first-order quantification and terms is mostly nonexistent. This can be seen for example in McCarty's realizability definition [10] or in our erasure maps [11,9]. We hope to investigate this topic further in the future.

The $\mathrm{in}(P)$ term intuitively denotes the proof of the fact that $\overline{P} \in_I \omega$. The ax term denotes proofs of computational equivalences.

The second group of the proof terms corresponds to the axioms of IZF_T:

$$\mathrm{inProp}(M) \mid \mathrm{inRep}(M) \mid \mathrm{eqProp}(M) \mid \mathrm{eqRep}(M) \mid \mathrm{emptyProp}(M) \mid \mathrm{emptyRep}(M)$$
$$\mathrm{pairProp}(M) \mid \mathrm{pairRep}(M) \mid \mathrm{unionProp}(M) \mid \mathrm{unionRep}(M) \mid \mathrm{sep}_{p,a,\boldsymbol{f}.\phi}\mathrm{Prop}(M)$$
$$\mathrm{sep}_{p,a,\boldsymbol{f}.\phi}\mathrm{Rep}(M) \mid \mathrm{powerProp}(M) \mid \mathrm{powerRep}(M) \mid \mathrm{omegaProp}(M) \mid \mathrm{omegaRep}(M)$$
$$\mathrm{repl}_{p,a,b,\boldsymbol{f}.\phi}\mathrm{Prop}(M) \mid \mathrm{repl}_{p,a,b,\boldsymbol{f}.\phi}\mathrm{Rep}(M) \mid \mathrm{ind}(M)$$

Intuitively, the Prop and Rep terms correspond to IZF_T axioms. For example, if M is a proof of $t \in_I P(u)$, then $\mathrm{powerProp}(M)$ is a proof of $t \subseteq u$ and if M is a proof of $t \subseteq u$, then $\mathrm{powerRep}(M)$ is a proof of $t \in_I P(u)$. As in our previous work [11,9,7], we adopt the convention of using axRep and axProp terms to tacitly mean all Rep and Prop terms, for ax being one of in, eq, empty, pair, union, sep, power, omega and repl. With this convention in mind, we can summarize the definition of the set-theoretic Prop and Rep terms as:

$$\mathrm{axProp}(M) \mid \mathrm{axRep}(M).$$

The third group of proof terms governs interaction between programs and sets.

$$\mathrm{inzRep}(M) \mid \mathrm{inzProp}(M) \mid \mathrm{insRep}(M) \mid \mathrm{insProp}(M) \mid \mathrm{eqpRep}(M)$$

Roughly, the proof terms $\mathrm{inzRep}(M), \mathrm{inzProp}(M)$ witness $\overline{0}$ being the empty set, the proof terms $\mathrm{insRep}(M), \mathrm{insProp}(M)$ witness $\overline{S(P)}$ being the same thing as $\overline{P} \cup \{\overline{P}\}$ and $\mathrm{eqpRep}(M)$ is used for a computational version of the Leibniz axiom. The type system in Section 3.3 will make these remarks precise.

Having finished describing the proof terms, we proceed to programs:

$$P ::= x \mid \lambda x.\ P \mid P\ Q \mid 0 \mid S(P) \mid \mathrm{casenat}(P, Q, x.R) \mid \mathrm{prog}(M)$$

This is a simple lambda calculus with natural numbers, which can be viewed as an extension of Gödel's system T in a version with iterator. The only new thing is the $\mathrm{prog}(M)$ term. The $\mathrm{prog}(M)$ term intuitively for any $M : t \in \omega$ denotes the natural number corresponding to t. This intuition will be validated by the reduction rules of P and the proof-theoretic properties of our framework. We will not use the $\mathrm{prog}(M)$ terms for M which are not proofs of $t \in \omega$.

The third part of the grammar generates the set terms:

$$t ::= a \mid \emptyset \mid \{t_1, t_2\} \mid \omega \mid P(t) \mid \bigcup t \mid S_{p,a,\boldsymbol{f}.\phi}(t, \boldsymbol{t}) \mid R_{p,a,b,\boldsymbol{f}.\phi}(t, \boldsymbol{t}) \mid \overline{P}$$

The $\overline{P}$ term intuitively denotes the member of ω corresponding to the program P.

The last part generates the formulas of D.

$$\phi ::= \bot \mid \mathrm{nat} \mid P \equiv Q \mid t \in_I u \mid t = u \mid t \in u \mid (p : \phi) \to \psi \mid (p : \phi) \wedge \psi \mid \phi \vee \psi \mid \forall a.\ \phi \mid \exists a.\ \phi$$

The formulas $(p : \phi) \to \psi$ and $(p : \phi) \wedge \psi$ are dependent versions of implication and conjunction. The variable p binds in ψ, which can mention p (inside of $\mathrm{prog}(M)$ terms). Traditional formulas $\phi \to \psi$ and $\phi \wedge \psi$ are defined as abbreviations for $(p : \phi) \to \psi$ and $(p : \phi) \wedge \psi$, where p is fresh.

There are two new atomic formulas which go beyond the dependent first-order logic. First, the inclusion of the formula/type nat among formulas makes it possible to use the machinery of the first-order logic to define types in P, via the Curry-Howard correspondence principle. Second, we allow reasoning about computational equivalence of programs by means of the formula $P \equiv Q$. The proof system in Section 3.3 should shed more light on these issues.

That programs and proofs terms are separate syntactic categories in our system is mostly a design choice. If one looked hard enough, one could find proof terms behaving similarly to programs from a computational point of view. However, the separation makes it possible to apply our framework easily to more complicated programming languages — it would be a simple exercise to extend P to incorporate pairs, lists, algebraic datatypes and other features met in modern functional programming languages.

3.2 The Reduction Relation

The reduction relation, denoted by $\to$, is deterministic and defined on proof terms and programs. It arises from reduction rules and evaluation contexts. In the reduction rules, we will use several proof terms corresponding to proofs of simple set-theoretic facts. Due to the space constraints, we do not present the full terms, but only state them as constants.

$$\mathrm{eqRefl} : \forall a.\ a = a \quad \mathrm{ii} : \forall a, b.\ a \in_I b \to a \in b \quad \mathrm{zz} : \overline{0} = \emptyset \quad \mathrm{ss} : \overline{S(P)} = \overline{P} \cup \{\overline{P}\}$$

So, for example, eqRefl stands for a proof term corresponding to the proof of $\forall a.\ a = a$. The term ss does not depend on P.

Now we can present the reduction rules. To avoid cluttering of the rules (and later also proofs), from now on we adopt the convention of using the _ character to denote the subterms which are of no interest to us and to the definition/proof in question.

The reduction rules are designed to make the Progress and Subject Reduction lemmas provable. The first group is standard [7] for constructive set theories:

$$(\lambda p.\ M)\ N \to M[p := N] \quad (\lambda *.\ M) * \to M \quad \mathrm{fst}(\langle M, \_\rangle) \to M \quad \mathrm{snd}(\langle \_, N\rangle) \to N$$
$$\mathrm{case}(\mathrm{inl}(M), p.\ N, p.\ \_) \to N[p := M] \quad \mathrm{case}(\mathrm{inr}(M), p.\ \_, p.\ O) \to O[p := M]$$
$$\mathrm{axProp}(\mathrm{axRep}(M)) \to M \quad \mathrm{ind}(M) \to \lambda *.\ M\ *\ (\lambda *.\lambda x.\ \mathrm{ind}(M)\ *)$$

There are two new rules governing the behavior of new proof terms:

$$\mathrm{in}(0) \to \mathrm{omegaRep}(\mathrm{inl}(\mathrm{zz})) \qquad \mathrm{in}(S(P)) \to \mathrm{omegaRep}(\mathrm{inr}([*, \langle \mathrm{in}(P), \mathrm{ss}\rangle]))$$

Furthermore, programs reduce as well, in an expected way:

$$(\lambda x.P)\ O \to O[x := P]$$
$$\mathrm{casenat}(0, Q, x.\_) \to Q \qquad \mathrm{casenat}(S(P), Q, x.R) \to R[x := \mathrm{casenat}(P, Q, x.R)]$$

Finally, we show the two rules governing the behavior of programs coming from set theory:

$$\mathrm{prog}(\mathrm{inRep}([*, \langle \mathrm{omegaRep}(\mathrm{inl}(\_)), \_\rangle])) \to 0$$
$$\mathrm{prog}(\mathrm{inRep}([*, \langle \mathrm{omegaRep}(\mathrm{inr}([*, \langle M, \_\rangle])), \_\rangle])) \to S(\mathrm{prog}(M)))$$

We call the reduction rules specified so far *atomic*. To extend these rules to all proof terms, we use the standard tool of evaluation contexts [12]. The evaluation contexts of P describe the call-by-need (lazy) evaluation order:

$$\begin{aligned}[\circ] ::= {} & \mathrm{fst}([\circ]) \mid \mathrm{snd}([\circ]) \mid \mathrm{case}([\circ], \_, \_) \mid \mathrm{in}([\circ]) \mid \mathrm{axProp}([\circ]) \mid [\circ]\ \_ \mid \mathrm{magic}([\circ]) \mid [\circ] * \\ & \mathrm{casenat}([\circ], \_, \_) \mid \mathrm{prog}([\circ]) \mid \mathrm{prog}(\mathrm{inRep}([\circ])) \mid \mathrm{prog}(\mathrm{inRep}([*, [\circ]])) \mid \\ & \mathrm{prog}(\mathrm{inRep}([*, \langle [\circ], \_\rangle])) \mid \mathrm{prog}(\mathrm{inRep}([*, \langle \mathrm{omegaRep}([\circ]), \_\rangle])) \mid \\ & \mathrm{prog}(\mathrm{inRep}([*, \langle \mathrm{omegaRep}(\mathrm{inr}([\circ])), \_\rangle])) \mid \\ & \mathrm{prog}(\mathrm{inRep}([*, \langle \mathrm{omegaRep}(\mathrm{inr}([*, [\circ]])), \_\rangle]))\end{aligned}$$

The reason for the large number of rules governing the behavior of prog(_) terms is that the subterms need to be evaluated in order to reach the form allowing the application of one of the atomic reduction rules.

We distinguish certain terms, listed below, as values.

$$\lambda\_.\ \_ \mid \mathrm{inr}(\_) \mid \mathrm{inl}(\_) \mid [\_, \_] \mid \langle \_, \_\rangle \mid \mathrm{axRep}(\_) \mid 0 \mid S(\_)$$

Definition 1. *We write $M \downarrow$ if the unique reduction sequence starting from M terminates. We write $M \downarrow v$, if v is the value M terminates at. We write $M \to^* N$ if M reduces to N in some number of steps.*

3.3 The Proof System of D

We now introduce the proof system for D. Contexts, denoted by Γ, are finite sets of pairs (p, ϕ), where p is a proof or program variable and ϕ is a formula. The *domain* of a context $\Gamma = p_1 : \phi_1, \ldots, p_n : \phi_n$, denoted by $\mathrm{dom}(\Gamma)$, is the set $\{p_1, \ldots, p_n\}$. The typing system is used to derive the judgments $\Gamma \vdash T : S$, read as "in environment Γ, T is of type S" or as "in environment Γ, T proves S".

The first group of rules corresponds to the first-order logic with dependent implications and conjunctions.

$$\frac{}{\Gamma, p : \phi \vdash p : \phi}\ p \notin \mathrm{dom}(\Gamma) \qquad \frac{\Gamma, p : \phi \vdash M : \psi}{\Gamma \vdash \lambda p.\ M : (p : \phi) \to \psi} \qquad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash \mathrm{magic}(M) : \phi}$$

$$\frac{\Gamma \vdash M : \phi}{\Gamma \vdash \lambda *\ .\ M : \forall a.\ \phi}\ a \notin FV(\Gamma) \qquad \frac{\Gamma \vdash M : (p : \phi) \to \psi \quad \Gamma \vdash N : \phi}{\Gamma \vdash M\ N : \psi[p := N]} \qquad \frac{\Gamma \vdash M : \forall a.\ \phi}{\Gamma \vdash M * : \phi[a := t]}$$

$$\frac{\Gamma \vdash M : \phi \quad \Gamma \vdash N : \psi[p := M]}{\Gamma \vdash \langle M, N\rangle : (p : \phi) \wedge \psi} \qquad \frac{\Gamma \vdash M : (p : \phi) \wedge \psi}{\Gamma \vdash \mathrm{fst}(M) : \phi} \qquad \frac{\Gamma \vdash M : (p : \phi) \wedge \psi}{\Gamma \vdash \mathrm{snd}(M) : \psi[p := \mathrm{fst}(M)]}$$

$$\frac{\Gamma \vdash M : \phi[a := t]}{\Gamma \vdash [*, M] : \exists a.\ \phi} \quad \frac{\Gamma \vdash M : \exists a.\ \phi \quad \Gamma, p : \phi \vdash N : \psi}{\Gamma \vdash \text{let } [*, p] := M \text{ in } N : \psi}\ a \notin FV(\Gamma, \psi)$$

$$\frac{\Gamma \vdash M : \phi}{\Gamma \vdash \text{inl}(M) : \phi \vee \psi} \quad \frac{\Gamma \vdash M : \psi}{\Gamma \vdash \text{inr}(M) : \phi \vee \psi} \quad \frac{\Gamma \vdash M : \phi \vee \psi \quad \Gamma, p : \phi \vdash N : \vartheta \quad \Gamma, p : \psi \vdash O : \vartheta}{\Gamma \vdash \text{case}(M, p.\ N, p.\ O) : \vartheta}$$

The second group of rules corresponds to the axioms of set theory.

$$\frac{\Gamma \vdash M : \phi_A(t, \boldsymbol{u})}{\Gamma \vdash \text{axRep}(M) : t \in_I t_A(\boldsymbol{u})} \quad \frac{\Gamma \vdash M : t \in_I t_A(\boldsymbol{u})}{\Gamma \vdash \text{axProp}(M) : \phi_A(t, \boldsymbol{u})}$$

$$\frac{\Gamma \vdash M : \forall c.\ (\forall b.\ b \in_I c \to \phi[a, \boldsymbol{f} := b, \boldsymbol{t}]) \to \phi[a, \boldsymbol{f} := c, \boldsymbol{t}]}{\Gamma \vdash \text{ind}(M) : \forall a.\ \phi[\boldsymbol{f} := \boldsymbol{t}]}$$

The third group of rules describes the typing system for the programs.

$$\frac{}{\Gamma \vdash 0 : \text{nat}} \quad \frac{\Gamma \vdash P : \text{nat}}{\Gamma \vdash S(P) : \text{nat}} \quad \frac{\Gamma \vdash P : \text{nat} \quad \Gamma \vdash Q : \phi \quad \Gamma, x : \phi \vdash R : \phi}{\Gamma \vdash \text{casenat}(P, Q, x.R) : \phi}$$

$$\frac{}{\Gamma \vdash \text{ax} : P \equiv P} \quad \frac{\Gamma \vdash \_ : P \equiv Q}{\Gamma \vdash \text{ax} : Q \equiv P} \quad \frac{\Gamma \vdash \_ : P \equiv Q}{\Gamma \vdash \text{ax} : R[P] \equiv R[Q]}\ R \text{ an arbitrary program}$$

$$\frac{}{\Gamma \vdash \text{ax} : P \equiv Q}\ P \to Q \text{ atomic} \quad \frac{\Gamma \vdash \_ : P \equiv Q \quad \Gamma \vdash \_ : Q \equiv R}{\Gamma \vdash \text{ax} : P \equiv R}$$

Finally, we present the rules glueing programs and sets together.

$$\frac{\Gamma \vdash P : \text{nat}}{\Gamma \vdash \text{in}(P) : \overline{P} \in_I \omega} \quad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash \text{inzRep}(M) : t \in_I \overline{0}} \quad \frac{\Gamma \vdash M : t \in_I \overline{0}}{\Gamma \vdash \text{inzProp}(M) : \bot}$$

$$\frac{\Gamma \vdash M : t \in_I \overline{P} \cup \{\overline{P}\}}{\Gamma \vdash \text{insRep}(M) : t \in_I \overline{S(P)}} \quad \frac{\Gamma \vdash M : t \in_I \overline{S(P)}}{\Gamma \vdash \text{insProp}(M) : t \in_I \overline{P} \cup \{\overline{P}\}}$$

$$\frac{\Gamma \vdash M : t \in_I \overline{P} \quad \Gamma \vdash \_ : P \equiv Q}{\Gamma \vdash \text{eqpRep}(M) : t \in_I \overline{Q}} \quad \frac{\Gamma \vdash M : t \in \omega}{\Gamma \vdash \text{prog}(M) : \text{nat}}$$

$$\frac{\Gamma \vdash M : t \in \omega \quad \Gamma \vdash N : u \in \omega \quad \Gamma \vdash O : t = u}{\Gamma \vdash \text{ax} : \text{prog}(M) \equiv \text{prog}(N)}$$

It is straightforward to show that $\mathrm{D} \vdash \mathrm{IZF}_R$. The standard properties of proof systems and programming languages: Inversion, Canonical Forms, Substitution Lemma, Weakening, Subject Reduction and Progress are proved using standard techniques, presented for example in [12,13] and applied to set theories in [7]. Although the normalization proof does involve some interesting twists, due to the space constraints we are forced to omit it. We prefer to give a more detailed account of Section 4, as the material presented there is much more novel and less established. We only remark that we use realizability as a proof technique, in a manner similar to [14].

Theorem 1 (Normalization, ZF + Con(ZF)). *If* $\vdash T : S$, *then* $T \downarrow$.

Corollary 1. *D is consistent.*

4 Mathematics

In this section, we shall see how D brings sets and programs together and allows them to interact. We will state and prove theorems using the properties and expressive power of D. However, we prefer to think of D as one of many possible axiomatizations of what could be termed *dependent mathematics*. We hope that our theorems can stand on their own and we remain hopeful for better axiomatizations to appear in the future.

Before we delve into proofs, let us ponder for a second about formulas in D. The formulas are made of standard first-order and set-theoretic features, but they can also use proof variables inside of $\mathrm{prog}(M)$ terms. The same comment applies to programs and set terms. The unconstrained usage of these proof variables, however, does not present immediate benefits, while it significantly complicates notation. For this reason, we introduce a notion of *supported* terms.

Definition 2. *A term T (which might be a program, a set term or a formula) is* supported *by a context Γ and proof trees $T_1, \ldots, T_n$, if $FV(T) \subseteq \mathrm{dom}(\Gamma)$ and if for any subterm S of T such that $S = \mathrm{prog}(M)$, there is i such that:*

- *For some t, T_i is a proof tree of $\Gamma \cup FV_t(S) \vdash M : t \in \omega$, and*
- *Any term U in T_i is supported by $\Gamma \cup FV_t(U)$ and $\{T_1, ..., T_n\} \setminus \{T_i\}$,*

where $FV_t(S)$ is the context consisting of all free variables of S bound in T by dependent implications and conjuctions. For example, for $T = (p : \phi) \wedge \mathrm{prog}(p)$ and $S = \mathrm{prog}(p)$, $FV_t(S) = \{(p, \phi)\}$.

The weight *of a supported term is the number of proof trees in its support.*

Intuitively, in a supported term, all $\mathrm{prog}(M)$ terms are typed.[1] For terms T supported by $\Gamma = p_1 : \phi_1, \ldots, p_n : \phi_n$ such that the free first-order variables of Γ are $\boldsymbol{a}$, we will use the notation $T(\boldsymbol{a}, \boldsymbol{p} : \boldsymbol{\psi})$. Note that this notation agrees with the standard first-order logic convention, as it (informally) says that the free variables of T are among $\boldsymbol{a}, \boldsymbol{p} : \boldsymbol{\psi}$.

Convention 2. *From now on, all formulas, set terms and programs we use are supported. We write $(p, \phi) \in FV(T)$ if (p, ϕ) is in a supporting context of T.*

Although we will rarely mention supporting contexts and proof trees explicitly, the reader should always assume that they are implicitly carried around.

If $(p, \phi) \in FV(T)$ (or $a \in FV(T)$), we write $T \equiv T[p : \phi]$ (or $T \equiv T[a]$), to mark all occurences of p (or a) in T. We read $T \equiv T[p : \phi]$ as "T is written as $T[p : \phi]$". With this notation, $T[M]$ denotes $T[p := M]$ and $T[t]$ denotes $T[a := t]$, respectively.

We also restrict the possible dependencies in formulas and their intensionality. Again, the reason is that full generality does not seem to be useful and that it is detrimental to some of the developments. The restriction amounts to allowing only first-order dependencies and extensional terms and formulas. Formally:

[1] We could have as well avoided supported terms completely by incorporating the notion into the typing system, at the price of vastly bloating proofs and the number of proof rules. Lemma 2 could then be proved by simple induction on proof trees.

Definition 3. *We call a term/formula* first-order*, if it does not contain any* $\mathrm{prog}(M)$ *terms. We call a formula* $\phi(\boldsymbol{a}, \boldsymbol{p} : \boldsymbol{\psi})$ flat*, if all* $\boldsymbol{\psi}$ *are first-order and if for any subformula* $(p : \phi_1) \bigoplus \phi_2$*, where* $\bigoplus \in \{\wedge, \to\}$*,* ϕ_1 *is first-order. We call a formula* extensional*, if it does not contain any* $\in_I$ *relational symbols. Extensional formulas may contain computational equivalences.*

Convention 3. *From now on, all terms and formulas we consider are flat and extensional.*

Since D is a dependent framework, proof terms and proofs play an essential role in the developments. There is no established tradition of presenting such developments in an informal mathematical discourse and our presentation is but a try. We believe that as importance of dependent mathematics will grow, a well-established discourse will evolve. For now, from the reader's point of view, the most important addition to the statements of set theory is a new judgment $M : \phi$, read as "M is a proof of ϕ" or "M proves ϕ", with its formal counterpart $\Gamma \vdash M : \phi$, where Γ implicitly contains all current assumptions and all supporting contexts.

We adopt three more convenient notational conventions. We write $\forall a.\ (p : a \in b) \to \phi$ as $\forall p : a \in b.\ \phi$ and similarly $\exists a.\ (p : a \in b) \wedge \phi$ as $\exists p : a \in b.\ \phi$. Moreover, since the meaning of the dependent conjunction in the world of programs is much closer to that of the (strong) existential quantifier, we will sometimes use the notation $\Sigma n : \phi.\ \psi$ to stand for $(n : \phi) \wedge \psi$. Finally, $\{p : a \in t \mid \phi\}$ stands for the set S such that $a \in S \leftrightarrow p : a \in t \wedge \phi$, existing due to the Separation axiom.

We start with the familiar Extensionality and Leibniz axioms.

Lemma 1 (Extensionality and Leibniz Axiom). $\forall a, b.\ a = b \leftrightarrow \forall c.\ c \in a \leftrightarrow c \in b$*. Moreover, for any term* t *and a formula* ϕ*:*

$$\forall a, b.\ a = b \to t[c := a] = t[c := b] \qquad \textit{and} \qquad a = b \to \phi[c := a] \to \phi[c := b]$$

Proof. Just as in [9], using the assumption of extensionality of t, ϕ on the way.

We next prove a technical lemma of crucial importance for further developments.

Lemma 2 (Proof Irrelevance). *Suppose* $N : \Psi$ *and* $O : \Psi$*. If* $(p, \Psi) \in FV(\phi)$*, then* $\phi[p := N] \leftrightarrow \phi[p := O]$*. Furthermore, if* $(p, \phi) \in FV(t)$*, then* $t[p := N] = t[p := O]$*. Finally, if* $(p, \phi) \in FV(P)$*, then* $P[p := N] \equiv P[p := O]$*.*

Proof First let us write down the claim more formally. Suppose $\Gamma \vdash N : \Psi$, $\Gamma \vdash O : \Psi$ and suppose $\Gamma, p : \Psi$ is a supporting context of ϕ, t and P, where $p \notin FV(\Gamma)$. Then $\Gamma \vdash \_ : \phi[p := N] \leftrightarrow \phi[p := O]$, $\Gamma \vdash \_ : t[p := N] = t[p := O]$ and $\Gamma \vdash_: P[p := N] \equiv P[p := O]$.

We proceed by lexicographical induction on the pair ("the weight of T", "the structural complexity of T"), where T is one of ϕ, t and P:

– Suppose P is $\mathrm{prog}(M)$. By the definition of supported programs, for some s, $\Gamma, p : \Psi \vdash M : s \in \omega$. Since the weight of s is smaller than the weight of P, by the induction hypothesis $\Gamma \vdash \_ : s[p := N] = s[p := O]$. By the Substitution Lemma, $\Gamma[p := N] \vdash M[p := N] : s[p := N] \in \omega$ and $\Gamma[p := O] \vdash M[p :=$

$O] : s[p := O] \in \omega$. Since $p \notin FV(\Gamma)$, $\Gamma \vdash M[p := N] : s[p := N] \in \omega$ and $\Gamma \vdash M[p := O] : s[p := O] \in \omega$. The following proof tree shows the claim, where T, S and U are proof trees of $\Gamma[p:=N] \vdash M[p:=N] : s[p := N] \in \omega$, $\Gamma[p := O] \vdash M[p := O] : s[p := O] \in \omega$ and $\Gamma \vdash \_ : s[p := N] = s[p := O]$, respectively.

$$\frac{T \quad S \quad U}{\Gamma \vdash \mathrm{ax} : \mathrm{prog}(M[p := N]) \equiv \mathrm{prog}(M[p := O])}$$

- For the rest of programs, the claim follows trivially, due to the definition and properties of computational equivalence $\equiv$.
- For the set terms, the claim also follows easily, due to the Leibniz axiom.
- For the formulas, the atomic cases follow easily by Extensionality and the Leibniz Axiom. The only interesting cases are the dependent formulas.
- Suppose ϕ is $(q : \phi_1) \to \phi_2$. Since ϕ is flat, ϕ_1 is first-order, so $p \notin FV(\phi_1)$. It is not difficult to see that $\Gamma, p : \Psi, q : \phi_1$ supports ϕ_2 (with the proof tress inherited from ϕ) and that the weight of ϕ_2 stays the same, so since ϕ_2 is structurally smaller tha ϕ, by the induction hypothesis $\Gamma, q : \phi_1 \vdash \_ : \phi_2[p := N] \leftrightarrow \phi_2[p := O]$. Therefore we can easily derive $\Gamma, \_ : \phi_1 \to \phi_2[p := N], q : \phi_1 \vdash \_ : \phi_2[p := O]$, from which the claim follows.
- Suppose ϕ is $(q : \phi_1) \wedge \phi_2$. Just as in the previous case, it is not difficult to see that $\Gamma, p : \Psi, q : \phi_1$ supports ϕ_2 and that its weight is the same as the weight of ϕ, so by the induction hypothesis, $\Gamma, q : \phi_1 \vdash \_ : \phi_2[p := N] \leftrightarrow \phi_2[p := O]$. Let $\Gamma' = \Gamma, r : (q : \phi_1) \wedge \phi_2[p := N]$. By Weakening, $\Gamma', q : \phi_1 \vdash \_ : \phi_2[p := N] \leftrightarrow \phi_2[p := O]$, so by the Substitution Lemma $\Gamma' \vdash \_ : \phi_2[p := N][q := \mathrm{fst}(r)] \leftrightarrow \phi_2[p := O][q := \mathrm{fst}(r)]$ .Since $\Gamma' \vdash \mathrm{fst}(r) : \phi_1$ and $\Gamma' \vdash \mathrm{snd}(r) : \phi_2[p := N][q := \mathrm{fst}(r)]$, the claim easily follows. □

From now on, we work in D. In other words, all lemmas and theorems apart from these labelled as schemas, have their formal counterparts in D.

Definition 4. *P_0 is the canonical proof of $0 \in \omega$. $P_S(n, q)$ is the canonical proof of $S(n) \in \omega$, given $q : n \in \omega$. Formally:*

$$P_0 \equiv \mathrm{ii} \ * \ * \ \mathrm{omegaRep}(\mathrm{inl}(\mathrm{eqRefl} \ *))$$
$$P_S(n, q) \equiv \mathrm{ii} \ * \ * \ \mathrm{omegaRep}(\mathrm{inr}([*, \langle q, \mathrm{eqRefl} \ * \rangle]))$$

Now we can derive our first truly new set-theoretical theorem: a dependent induction principle. Roughly speaking, just as it is sufficient in standard proofs by mathematical induction to consider only the cases for zero and successor, in the dependent case it suffices to consider only the canonical proofs of membership of respective sets in ω.

Theorem 4 (Dependent Induction Schema). *For any ϕ such that $(p, n \in \omega) \in FV(\phi)$, let $\phi \equiv \phi[p, n]$. If $\phi[P_0, 0]$ and $\forall q : n \in \omega.\ \phi[q, n] \to \phi[P_S(n, q), n \cup \{n\}]$, then $\forall p : n \in \omega.\ \phi[p, n]$.*

Proof. Consider the set $A \equiv \{p : n \in \omega \mid \phi[p, n]\}$.

First, by $\phi[P_0, 0]$, $0 \in A$. Second, take any $n \in A$. Then there is $p : n \in \omega$ such that $\phi[p, n]$. Therefore $\phi[P_S(n, p), n \cup \{n\}]$. Since $P_S(n, p) : n \cup \{n\} \in \omega$,

$n \cup \{n\} \in A$. Thus A contains 0 and is closed under successor, so as ω is the smallest set containing 0 and closed under successor, $\omega \subseteq A$.

So take any $p : n \in \omega$. Since $\omega \subseteq A$, $n \in A$, so there is r such that $r : (q : n \in \omega) \wedge \phi[q, n]$. Therefore $\mathrm{fst}(r) : n \in \omega$ and $\mathrm{snd}(r) : \phi[\mathrm{fst}(r), n]$. By Proof Irrelevance, $\phi[\mathrm{fst}(r), n] \leftrightarrow \phi[p, n]$. Therefore we also have $\phi[p, n]$, which shows the claim.

Lemma 3 (Program schema). *For all programs P, Q, if $P \equiv Q$, then $\overline{P} = \overline{Q}$.*

Proof. Suppose $P \equiv Q$ and take any $A \in_I \overline{P}$. Then $A \in_I \overline{Q}$, so $A \in \overline{Q}$. On the other hand, suppose $A \in_I \overline{Q}$. Since $\equiv$ is symmetric, the claim easily follows.

Lemma 4 (Program schema). *For any $q : n \in \omega$, $\mathrm{prog}(P_S(n, q)) \to^* S(\mathrm{prog}(q))$.*

Lemma 5. *For all $p : n \in \omega$, $\overline{\mathrm{prog}(p)} = n$.*

Proof. We proceed by dependent induction.

- Case $P_0 : \emptyset \in \omega$. We need to show $\overline{\mathrm{prog}(P_0)} = \emptyset$. Since $\mathrm{prog}(P_0) \to 0$, $\mathrm{prog}(P_0) \equiv 0$. By Lemma 3, $\overline{\mathrm{prog}(P_0)} = \overline{0}$. Since $\mathrm{zz} : \overline{0} = \emptyset$, the claim follows.
- Given $p : n \in \omega$ such that $\overline{\mathrm{prog}(p)} = n$, we need to show $\overline{\mathrm{prog}(P_S(p))} = n \cup \{n\}$. By Lemma 4, $\mathrm{prog}(P_S(p)) \to^* S(\mathrm{prog}(p))$, so $\mathrm{prog}(P_S(p)) \equiv S(\mathrm{prog}(p))$. By Lemma 3, $\overline{\mathrm{prog}(P_S(p))} = \overline{S(\mathrm{prog}(p))}$. We also have $\overline{S(\mathrm{prog}(p))} = \mathrm{prog}(p) \cup \{\overline{\mathrm{prog}(p)}\}$. Since $\overline{\mathrm{prog}(p)} = n$, $\mathrm{prog}(p) \cup \{\overline{\mathrm{prog}(p)}\} = n \cup \{n\}$, thus the claim follows. □

Since D normalizes, it is easy [7] to derive the disjunction, numerical existence and term existence properties for D. However, D offers also a much better choice — we can state and prove the numerical existence property *inside* D:

Lemma 6. $\forall a \in \omega.\ (\phi(a) \to \Sigma n : nat.\ \phi(\overline{n}))$.

Proof. Take any $p : a \in \omega$ such that $\phi(a)$. Set $n = \mathrm{prog}(p)$. We need to show $\phi(\overline{\mathrm{prog}(p)})$. By Lemma 5, $\overline{\mathrm{prog}(p)} = a$, so since we have $\phi(a)$, the claim follows by the Leibniz axiom.

Corollary 2 (Numerical Existence Property).

$$(\exists a \in \omega.\ \phi(a)) \to \Sigma n : \mathrm{nat}.\ \phi(\overline{n})$$

Definition 5. *For $M : \exists a \in \omega.\ \phi(a)$ we define* $\mathrm{nep}(M)$ *to be the term we obtain from Corollary 2:*

$$\mathrm{nep}(M) : \Sigma n : \mathrm{nat}.\ \phi(\overline{n})$$

Note that Corollary 2 states exactly what the Numerical Existence Property usually says on a metalevel: if $T \vdash \exists a \in \omega.\ \phi(a)$, then there is a numeral n such that $T \vdash \phi(n)$. Thanks to normalization and the properties of D, we know that $\overline{\mathrm{fst}(\mathrm{nep}(M))}$ is such a numeral.

It would not be difficult to extend the system with booleans and internalize the disjunction property in a similar way. The term existence property, however, due to contradictions lurking around the corner [14], does not seem to be easily internalizable.

The Numerical Existence Property is just the beginning. We can also state and prove program extraction meaningfully:

Theorem 5 (Program Extraction).

$$(\forall x \in \omega \exists y \in \omega.\ \phi(x,y)) \to \Sigma f : \text{nat} \to \text{nat}.\ \forall q : x \in \omega.\ \phi(x, \overline{f\ \text{prog}(q)})$$

Proof. Suppose $p : \forall x \in \omega \exists y \in \omega.\ \phi(x,y)$. Define f as follows. f takes $n : \text{nat}$ and returns $\text{fst}(\text{nep}(p\ *\ (\text{ii}\ *\ *\ \text{in}(n))))$. To show correctness, take any $q : x \in \omega$. We know there is $y \in \omega$ such that $\phi(x,y)$. By the definition of f and properties of nep, $f\ \text{prog}(q)$ is the $m : \text{nat}$ such that $\phi(\overline{\text{prog}(q)}, \overline{m})$. Since $\overline{\text{prog}(q)} = x$, we also have $\phi(x, \overline{m})$, so the claim follows.

The following version of the Axiom of Choice shows that the computation influences significantly the set-theoretic part of D. An interesting question, which we leave open, is whether stronger forms of AC, such as Dependent Choice, could be proved as well.

Theorem 6 ($\mathbf{AC}_{\omega,\omega}$). *If $\forall x \in \omega \exists y \in \omega.\ \phi(x,y)$ then there is a function $f : \omega \to \omega$ such that for all $x \in \omega$, $\phi(x, f(x))$.*

Proof. Suppose $p : \forall x \in \omega \exists y \in \omega.\ \phi$. Define:

$$n(q) \equiv \text{fst}(\text{nep}(p * (\text{ii} * * \text{in}(\text{prog}(q))))) \qquad \text{and} \qquad f \equiv \{z \mid \exists q : x' \in \omega.\ z = (x', \overline{n(q)})\}$$

We first show that f is a function. Take any $q : x \in \omega$. Let $y = \overline{n(q)}$. It is easy to see that $n(q) : \text{nat}$, so $y \in \omega$. And obviously $(x,y) \in f$.

Now, take any y' such that $(x,y') \in f$. We need to show that $y' = y$. Since $(x,y') \in f$, we know that there is $q' : x' \in \omega$ such that $(x,y') = (x', \overline{n(q')})$. Therefore, $x = x'$ and $y' = \overline{n(q')}$. Since $q : x \in \omega, q' : x' \in \omega$ and $x = x'$, $\text{prog}(q) \equiv \text{prog}(q')$, therefore $n(q) \equiv n(q')$ and $\overline{n(q)} = \overline{n(q')}$, so $y = y'$.

Finally, take any $q : x \in \omega$. We need to show that $\phi(x, f(x))$. In other words, that $\phi(x, \overline{n(q)})$. This follows by Lemma 4, the Leibniz axiom and the properties of nep.

Corollary 3. *In D, Dedekind real numbers and Cauchy real numbers are the same. Furthermore, D is stronger than IZF_R, as the first part of the claim does not hold in IZF_R.*

Proof. The first part of the claim follows by Proposition 3.21 in [15]. We learned the second part of the claim by personal communication with Lubarsky; it follows easily from the first part of the claim and results of [16]. We were also informed that it easily follows from results of [17] as well.

We conclude with the program and the claim we started with.

Lemma 7. *Let* $\text{f} = \lambda x.\ \text{casenat}(x, 0, y.S(y))$. *Then for all $p : n \in \omega$, we have $f\ (\text{prog}(p)) \equiv \text{prog}(p)$.*

Proof. We proceed by dependent induction on $p : n \in \omega$. For $P_0 : 0 \in \omega$, we have $f\ (\text{prog}(P_0)) \to^* 0$ and also $\text{prog}(P_0) \to^* 0$, so the claim follows. Suppose we have the claim for $q : n \in \omega$. We need to show it for $P_S(n,q) : S(n) \in \omega$. By Lemma 4, $\text{prog}(P_S(n,q)) \to^* S(\text{prog}(q))$, so $f\ \text{prog}(P_S(n,q)) \to^* S(\text{prog}(q))$. The claim follows by reflexivity of $\equiv$.

5 Conclusion

We have presented a new foundational framework, unifying sets and programs and showed how the combination enriches both worlds. We believe this work is but a first step into the realm of dependent set theory. The fact that it is possible now to clearly delineate "concrete" computational objects from their set-theoretic counterparts, yet allow them to interact in a unified setting, raises numerous interesting questions. For example, how exactly does tinkering with the programming language and its type system influence set theory? How far inside the realm of sets can we go to discover the computational content hiding in the proofs?

A more concrete question concerns the formulation of D. We first proved the normalization theorem for D and later restricted the system for the purpose of dependent mathematics. While this approach allowed us to prove the normalization theorem for a much stronger system, it would likely be an obstacle to implementing D. Is there a nicer formulation of the restricted part of D?

We are hopeful to see answers to these questions.

5.1 Related Work

Dependent type theories, dating back to deBruijn's Automath, implemented in many systems [3,2,18], are a foundational setting which also integrates logic with programs. Nuprl [2], in particular, is close to our D, since its programming language is defined *before* the logic and can be extended freely.

[14] presents a dependent set theory IZF_D, resulting by extending IZF_R with dependent implications, conjuctions and restricted Σ-types. IZF_D does not have nice proof-theoretic properties, such as Subject Reduction, and it does not support the post-construction-verification paradigm. It also does not have the Numerical Existence Property, not to mention capability to internalize it.

Howe [19] combines a programming language with set theory to provide a model for extensions of Nuprl. He does not provide a formal system to axiomatize his model. Map theory [20] integrates programs and sets in a setting stronger than ZFC. Since it does not have a nice axiomatization from the proof-theoretical point of view, it does not support the correct-by-construction paradigm.

Constructive set theories have also been interpreted in computational frameworks such as type theory [21] and deduction modulo [22]. [23,24] investigate linear set theories. These investigations share the foundational character of our work. Their goals and results are different from ours, however.

We are grateful to anonymous reviewers for their helpful comments.

References

1. Gordon, M., Melham, T.: Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic. Cambridge University Press, Cambridge (1993)
2. Constable, R.L.: The structure of Nuprl's type theory. In: Logic of Computation. Series F: Computer and Systems Sciences, vol. 157, pp. 123–156. Springer, Heidelberg (1997)

3. The Coq Development Team: The Coq Proof Assistant Reference Manual – Version V8.0 (2004)
4. Benl, H., Berger, U., Schwichtenberg, H., et al.: Proof theory at work: Program development in the Minlog system. In: Bibel, W., Schmitt, P.G. (eds.) Automated Deduction, vol. II, pp. 41–71. Kluwer, Dordrecht (1998)
5. Abrial, J.R.: The B-book: assigning programs to meanings. Cambridge University Press, New York (1996)
6. Constable, R., Moczydłowski, W.: Extracting Programs from Constructive HOL Proofs via IZF Set-Theoretic Semantics. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS, vol. 4130, pp. 162–176. Springer, Heidelberg (2006)
7. Moczydłowski, W.: Investigations on Sets and Types. Ph.D thesis, Cornell University (2007)
8. Myhill, J.: Some properties of intuitionistic Zermelo-Fraenkel set theory. In: Cambridge Summer School in Mathematical Logic, vol. 29, pp. 206–231. Springer, Heidelberg (1973)
9. Moczydłowski, W.: A Normalizing Intuitionistic Set Theory with Inaccessible Sets. Logical Methods in Computer Science 3 (2007)
10. McCarty, D.: Realizability and Recursive Mathematics. D.Phil. Thesis, University of Oxford (1984)
11. Moczydłowski, W.: Normalization of IZF with Replacement. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 516–530. Springer, Heidelberg (2006)
12. Pierce, B.C.: Types and Programming Languages. MIT Press, Cambridge (2002)
13. Sørensen, M., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Elsevier, Amsterdam (2006)
14. Moczydłowski, W.: A Dependent Set Theory. In: Proceedings of LICS 2007, pp. 23–34. IEEE Computer Society Press, Los Alamitos (2007)
15. Aczel, P., Rathjen, M.: Notes on constructive set theory. Technical Report 40, Institut Mittag-Leffler (The Royal Swedish Academy of Sciences) (2000/2001)
16. Lubarsky, R.S.: On the Cauchy Completeness of the Constructive Cauchy Reals. Electron. Notes Theor. Comput. Sci. 167, 225–254 (2007)
17. Fourman, M.P., Hyland, J.: Sheaf models for analysis. In: Applications of Sheaves, pp. 280–301. Springer, Heidelberg (1979)
18. Pollack, R.: The Theory of LEGO:A Proof Checker for the Extended Calculus of Constructions. Ph.D thesis, Department of Computer Science, University of Edinburgh (1995)
19. Howe, D.J., Stoller, S.D.: An Operational Approach to Combining Classical Set Theory and Functional Programming Languages. In: Hagiya, M., Mitchell, J.C. (eds.) TACS 1994. LNCS, vol. 789, pp. 36–55. Springer, Heidelberg (1994)
20. Grue, K.: Map theory. Theor. Comput. Sci. 102(1), 1–133 (1992)
21. Aczel, P.: The type theoretic interpretation of constructive set theory. In: Logic Colloquium 1977, pp. 55–66. North Holland, Amsterdam (1978)
22. Dowek, G., Miquel, A.: Cut elimination for Zermelo's set theory (2006); Manuscript, available from the web pages of the authors
23. Shirahata, M.: Linear Set Theory. Ph.D thesis, Stanford University (1994)
24. Terui, K.: Light affine set theory: A naive set theory of polynomial time. Studia Logica 77(1), 9–40 (2004)

Product-Free Lambek Calculus Is NP-Complete

Yury Savateev*

Department of Mathematical Logic, Faculty of Mechanics and Mathematics,
Moscow State University, Moscow, 119991, Russia

Abstract. In this paper we prove that the derivability problems for product-free Lambek calculus and product-free Lambek calculus allowing empty premises are NP-complete. Also we introduce a new derivability characterization for these calculi.

Introduction

Lambek calculus L was first introduced in [4]. It can be used for describing natural language syntax and for specifying formal languages (sets of finite words over a finite alphabet). This is done in the framework of categorial grammars, where all language-specific information is put in a lexicon and the derivation rules are the same for all languages. Therefore, the derivability problem (whether or not a given sequent is derivable) for Lambek calculus is important for applications. The expressive power of Lambek categorial grammars equals that of context-free grammars (see [6]).

Lambek calculus uses syntactic types that are built from primitive types using three binary connectives: multiplication, left division, and right division. Natural fragments of Lambek calculus are the product-free Lambek calculus $\mathrm{L}(\backslash, /)$, which does not use multiplication, and the unidirectional Lambek calculi, which have only one connective left: a division (left or right). Categorial grammars based on these fragments have the same expressive power as grammars based on the full version (the equivalence of unidirectional Lambek categorial grammars and context-free grammars was proved in [2]).

For the non-associative variant of Lambek calculus the derivability can be checked in polynomial time as shown in [3] (for the product-free fragment of the non-associative Lambek calculus this was proved already in [1]).

In [5] NP-completeness was proved for the derivability problem for full associative Lambek calculus. In [7] there was presented a polynomial algorithm for its unidirectional fragments.

We show that the classical satisfiability problem *SAT* is polynomial time reducible to the $\mathrm{L}(\backslash, /)$-derivability problem and thus $\mathrm{L}(\backslash, /)$ is NP-complete.

* This research was supported in part by the Russian Foundation for Basic Research grant 08-01-00399.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 380–394, 2009.

1 Product-Free Lambek Calculus

Product-free Lambek calculus $\mathrm{L}(\backslash, /)$ can be constructed as follows. Let $\mathbf{P} = \{p_0, p_1, \ldots\}$ be a countable set of what we call *primitive types*. Let Tp be the set of *types* constructed from primitive types with two binary connectives $/$, $\backslash$. We will denote primitive types by small letters (p, q, r, ...) and types by capital letters (A, B, C, ...). By capital greek letters (Π, Γ, Δ,...) we will denote finite (possibly empty) sequences of types. Expressions like $\Pi \to A$, where Π is not empty, are called *sequents*.

Axioms and rules of $\mathrm{L}(\backslash, /)$:

$$A \to A, \qquad \frac{\Phi \to B \quad \Gamma B \Delta \to A}{\Gamma \Phi \Delta \to A}\ (\mathrm{CUT}),$$

$$\frac{\Pi A \to B}{\Pi \to (B/A)}\ (\to /), \qquad \frac{\Phi \to A \quad \Gamma B \Delta \to C}{\Gamma (B/A) \Phi \Delta \to C}\ (/ \to),$$

$$\frac{A \Pi \to B}{\Pi \to (A \backslash B)}\ (\to \backslash), \qquad \frac{\Phi \to A \quad \Gamma B \Delta \to C}{\Gamma \Phi (A \backslash B) \Delta \to C}\ (\backslash \to),$$

(Here Γ and Δ can be empty.)

In this paper we will consider two calculi — $\mathrm{L}(\backslash, /)$ and $\mathrm{L}^*(\backslash, /)$, called product-free Lambek calculus allowing empty premises. In $\mathrm{L}^*(\backslash, /)$ we allow the antecedent of a sequent to be empty.

It can be shown that in these calculi every derivable sequent has a cut-free derivation where all instances of the axiom are of the form $p \to p$ where $p \in \mathbf{P}$.

2 Reduction from *SAT*

Let $c_1 \wedge \ldots \wedge c_m$ be a Boolean formula in conjunctive normal form with clauses $c_1 \ldots c_m$ and variables $x_1 \ldots x_n$. The reduction maps the formula to a sequent, which is derivable in $\mathrm{L}(\backslash, /)$ (and in $\mathrm{L}^*(\backslash, /)$) if and only if the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable.

For any Boolean variable x_i let $\neg_0 x_i$ stand for the literal $\neg x_i$ and $\neg_1 x_i$ stand for the literal x_i.

Note that $\langle t_1, \ldots, t_n \rangle \in \{0,1\}^n$ is a satisfying assignment for the Boolean formula $c_1 \wedge \ldots \wedge c_m$ if and only if for every $j \leq m$ there exists $i \leq n$ such that the literal $\neg_{t_i} x_i$ appears in the clause c_j (as usual, 1 stands for "true" and 0 stands for "false").

Let $p_i^j, q_i^j, a_i^j, b_i^j, c_i^j, d_i^j; 0 \leq i \leq n, 0 \leq j \leq m$ be distinct primitive types from $\mathbf{P}$.

We define the following families of types:

$$
\begin{aligned}
G^0 &\rightleftharpoons (p_0^0 \backslash p_n^0)\\
G^j &\rightleftharpoons (q_n^j / ((q_0^j \backslash p_0^j) \backslash G^{j-1})) \backslash p_n^j\\
G &\rightleftharpoons G^m\\
A_i^0 &\rightleftharpoons (a_i^0 \backslash p_i^0)\\
A_i^j &\rightleftharpoons (q_i^j / ((b_i^j \backslash a_i^j) \backslash A_i^{j-1})) \backslash p_i^j\\
A_i &\rightleftharpoons A_i^m\\
C_i^0 &\rightleftharpoons (c_i^0 \backslash p_i^0)\\
C_i^j &\rightleftharpoons (q_i^j / ((d_i^j \backslash c_i^j) \backslash C_i^{j-1})) \backslash p_i^j\\
C_i &\rightleftharpoons C_i^m\\
E_i^0(t) &\rightleftharpoons p_{i-1}^0\\
E_i^j(t) &\rightleftharpoons \begin{cases} q_i^j / (((q_{i-1}^j / E_i^{j-1}(t)) \backslash p_{i-1}^j) \backslash p_i^{j-1}), & \text{if } \neg_t x_i \text{ appears in } c_j\\ (q_{i-1}^j / (q_i^j / (E_i^{j-1}(t) \backslash p_i^{j-1}))) \backslash p_{i-1}^j, & \text{if } \neg_t x_i \text{ does not appear in } c_j \end{cases}\\
F_i^j(t) &\rightleftharpoons (E_i^j(t) \backslash p_i^j)\\
F_i(t) &\rightleftharpoons F_i^m(t)\\
H_i^0 &\rightleftharpoons p_{i-1}^0 \backslash p_i^0\\
H_i^j &\rightleftharpoons ((q_{i-1}^j / (q_i^j / H_i^{j-1})) \backslash p_{i-1}^j) \backslash p_i^j\\
H_i &\rightleftharpoons H_i^m\\
B_i^0 &\rightleftharpoons a_i^0\\
B_i^j &\rightleftharpoons q_{i-1}^j / (((b_i^j / B_i^{j-1}) \backslash a_i^j) \backslash p_{i-1}^{j-1})\\
B_i &\rightleftharpoons B_i^m \backslash p_{i-1}^m\\
D_i^0 &\rightleftharpoons c_i^0\\
D_i^j &\rightleftharpoons q_{i-1}^j / (((d_i^j / D_i^{j-1}) \backslash c_i^j) \backslash p_{i-1}^{j-1})\\
D_i &\rightleftharpoons D_i^m \backslash p_{i-1}^m.
\end{aligned}
$$

Let Π_i denote the following sequences of types:

$$(F_i(0)/(B_i \backslash A_i))\ H_i\ ((D_i \backslash C_i) \backslash F_i(1)).$$

Theorem 1. *The following statements are equivalent:*

1. $c_1 \wedge \ldots \wedge c_m$ *is satisfiable.*
2. $\mathrm{L}(\backslash, /) \vdash \Pi_1 \ldots \Pi_n \to G$
3. $\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \ldots \Pi_n \to G.$

This theorem will be proven in section 4.

3 Derivability Characterization

Let At be the set of *atoms* or *primitive types with superscripts*, $\{p^{(i)} | p \in \mathbf{P}, i \in \mathbb{Z}\}$. Let FS be the free monoid (the set of all finite strings) generated by elements of At. We will denote elements of FS by $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ and so on, by ε we will denote the empty string.

Consider two mappings:

$$t : \mathrm{FS} \to \mathbf{P}, t(\mathbb{A}p^{(i)}) = p; \quad d : \mathrm{FS} \to \mathbb{Z}, d(\mathbb{A}p^{(i)}) = i.$$

Let $\mathbb{A} \sqsubset \mathbb{B}$ denote that $\mathbb{A}$ is a strict prefix of $\mathbb{B}$ (i.e. there is $\mathbb{C} \neq \varepsilon \in \mathrm{FS}$ such that $\mathbb{B} = \mathbb{A}\mathbb{C}$). We will denote such $\mathbb{C}$ as $\mathbb{A} \backslash\!\!\backslash \mathbb{B}$. By $\mathbb{A} \sqsubseteq \mathbb{B}$ we will denote that either $\mathbb{A} \sqsubset \mathbb{B}$ or $\mathbb{A} = \mathbb{B}$. We can define in the usual way the following notions: $\min_{\sqsubset}$, $\max_{\sqsubset}$, $\inf_{\sqsubset}$, $\sup_{\sqsubset}$, $[\mathbb{A}, \mathbb{B}]_{\sqsubset}$, and $(\mathbb{A}, \mathbb{B}]_{\sqsubset}$.

For $\mathbb{A} \in \mathrm{FS}, \mathbb{A} \neq \varepsilon$ let $\mathcal{P}_{\mathbb{A}} = \{\mathbb{B} \mid \mathbb{B} \sqsubseteq \mathbb{A}, \mathbb{B} \neq \varepsilon\}$. The relation $\sqsubseteq$ is a total order on $\mathcal{P}_{\mathbb{A}}$.

Let α be a partial function on $\mathcal{P}_{\mathbb{A}}$. For each such function we can define the following:

$$\begin{gathered}
\mathbb{B} <_\alpha \mathbb{C} \Leftrightarrow \exists n \geq 1, \alpha^n(\mathbb{B}) = \mathbb{C}, \\
\mathbb{B} \leq_\alpha \mathbb{C} \Leftrightarrow \mathbb{B} <_\alpha \mathbb{C} \vee \mathbb{B} = \mathbb{C}, \\
\mu_\alpha^-(\mathbb{B}) = \min_{\sqsubset}(\mathbb{B}, \alpha(\mathbb{B})), \\
\mu_\alpha^+(\mathbb{B}) = \max_{\sqsubset}(\mathbb{B}, \alpha(\mathbb{B})), \\
\mathcal{F}_\alpha(\mathbb{B}) = \{\mathbb{C} \mid \mathbb{C} \leq_\alpha \mathbb{B}\}, \\
\nu_\alpha^-(\mathbb{B}) = \inf_{\sqsubset}(\mathcal{F}_\alpha(\mathbb{B})), \\
\nu_\alpha^+(\mathbb{B}) = \sup_{\sqsubset}(\mathcal{F}_\alpha(\mathbb{B})).
\end{gathered}$$

Consider two antiendomorphisms $(\cdot)^{\leftarrow}$ and $(\cdot)^{\rightarrow}$ on FS defined by

$$\begin{gathered}
(p^{(0)})^{\leftarrow} = p^{(-1)}, \quad (p^{(0)})^{\rightarrow} = p^{(1)}, \\
(p^{(i)})^{\leftarrow} = (p^{(i)})^{\rightarrow} = p^{(-i-sgn(i))}, \text{for } i \neq 0.
\end{gathered}$$

(A function $f\colon X \to X$ is an antiendomorphism if $\forall a, b \in X, f(ab) = f(b)f(a)$. In a free monoid it can be defined by its actions on the generators).

Consider $[\![\cdot]\!]\colon \mathrm{Tp} \to \mathrm{FS}$, a mapping from Lambek types to elements of the free monoid defined by

$$[\![p]\!] = p^{(0)}, \quad [\![(A/B)]\!] = [\![B]\!]^{\rightarrow}[\![A]\!], \quad [\![(A\backslash B)]\!] = [\![B]\!][\![A]\!]^{\leftarrow}.$$

Let us define φ — the partial function on $\mathcal{P}_{[\![A]\!]}$ that reflects the structure of the Lambek type A:

$$\varphi(\mathbb{A}) = \begin{cases} \inf_{\sqsubset}\{\mathbb{B} \mid \mathbb{A} \sqsubset \mathbb{B}, |d(\mathbb{B})| = |d(\mathbb{A})| - 1\}, & \text{if } d(\mathbb{A}) > 0; \\ \sup_{\sqsubset}\{\mathbb{B} \mid \mathbb{B} \sqsubset \mathbb{A}, |d(\mathbb{B})| = |d(\mathbb{A})| - 1\}, & \text{if } d(\mathbb{A}) < 0. \end{cases}$$

It can be easily shown that the following facts hold:

1. There is a unique $\mathbb{A}_0 \in \mathcal{P}_{[\![A]\!]}$ such that $d(\mathbb{A}_0) = 0$.
2. $\varphi(\mathbb{A})$ is defined for every $\mathbb{A} \neq \mathbb{A}_0$.
3. $\leq_\varphi$ is a partial order on $\mathcal{P}_{[\![A]\!]}$.
4. For every $i \in \mathbb{N}$ such that $i < |d(\mathbb{A})|$ there exists $\mathbb{B}$ such that $|d(\mathbb{B})| = i$ and $\mathbb{A} <_\varphi \mathbb{B}$, for instance $\mathbb{A} \leq_\varphi \mathbb{A}_0$.
5. If $\mathbb{A} \in [\mu^-_\varphi(\mathbb{B}), \mu^+_\varphi(\mathbb{B})]_\sqsubset$, then $\mathbb{A} \leq \varphi(\mathbb{B})$.

Suppose $\mathbb{A}, \mathbb{B} \in \mathcal{P}_{[\![A]\!]}$. There exists $\mathbb{C} \in \mathcal{P}_{[\![A]\!]}$ such that $\mathbb{A} \leq_\varphi \mathbb{C}$, $\mathbb{B} \leq_\varphi \mathbb{C}$, and for all $\mathbb{C}' \in \mathcal{P}_{[\![A]\!]}$ such that $\mathbb{A} <_\varphi \mathbb{C}'$ and $\mathbb{A} \leq_\varphi \mathbb{C}'$, $\mathbb{C} \leq_\varphi \mathbb{C}'$. Such $\mathbb{C}$ is called the φ-join of $\mathbb{A}$ and $\mathbb{B}$.

A set $\mathcal{G} \subset \mathcal{P}_{[\![A]\!]}$ is called φ-closed if there is no $\mathbb{A} \notin \mathcal{G}$ such that $\varphi(\mathbb{A}) \in \mathcal{G}$.

Let $\mathcal{N}_\mathbb{A} = \{\mathbb{B} \in \mathcal{P}_\mathbb{A} \mid d(\mathbb{B}) = 2i + 1, i \in \mathbb{Z}\}$.

Suppose we have a Lambek sequent $A_1 \dots A_n \to B$. Let

$$\mathbb{W} = [\![(\dots(B/A_n)/\dots)/A_1]\!] = [\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow}[\![B]\!].$$

Let π be a function on $\mathcal{P}_\mathbb{W}$, and ψ be a partial function defined by

$$\psi(\mathbb{A}) = \begin{cases} \pi(\mathbb{A}), & \text{if } \mathbb{A} \in \mathcal{N}_\mathbb{W}; \\ \varphi(\mathbb{A}), & \text{if } \mathbb{A} \notin \mathcal{N}_\mathbb{W} \text{ and } d(\mathbb{A}) \neq 0. \end{cases}$$

To characterize derivability of the sequent $A_1 \dots A_n \to B$ we shall use the following conditions, which we call proof conditions.

1. If $\mathbb{A} \in \mathcal{N}_\mathbb{W}$, then $\pi(\mathbb{A}) \notin \mathcal{N}_\mathbb{W}$ and $\pi^2(\mathbb{A}) = \mathbb{A}$ for all $\mathbb{A} \in \mathcal{P}_\mathbb{W}$.
2. $t(\pi(\mathbb{A})) = t(\mathbb{A})$.
3. $\mu^-_\pi(\mathbb{A}) \sqsubset \mu^-_\pi(\mathbb{B}) \Rightarrow \mu^+_\pi(\mathbb{A}) \sqsubset \mu^-_\pi(\mathbb{B}) \vee \mu^+_\pi(\mathbb{B}) \sqsubset \mu^+_\pi(\mathbb{A})$.
4. $\mathbb{A} \in \mathcal{N}_\mathbb{W} \Longrightarrow \mathbb{A} <_\psi \varphi(\mathbb{A})$ or equivalently $\forall \mathbb{A} \in \mathcal{P}_\mathbb{W}, \mathcal{F}_\varphi(\mathbb{A}) \subset \mathcal{F}_\psi(\mathbb{A})$.
5. $\mathbb{A} \notin \mathcal{N}_\mathbb{W} \wedge \mathbb{A} \neq \mathbb{A}_0 \Longrightarrow \exists \mathbb{B}(\mathbb{B} <_\psi \mathbb{A} \wedge \mathbb{B} \not<_\varphi \mathbb{A})$.

We will call $\mathcal{G} \subset \mathcal{P}_\mathbb{W}$ π-closed if for all $\mathbb{A} \in \mathcal{G}, \pi(\mathbb{A}) \in \mathcal{G}$. It is readily seen that if π satisfies proof conditions (1) and (3), then for every $\mathbb{A} \in \mathcal{N}_\mathbb{W}$, $[\mu^-_\pi(\mathbb{A}), \mu^+_\pi(\mathbb{A})]_\sqsubset$ and $\mathcal{P}_\mathbb{W} \setminus [\mu^-_\pi(\mathbb{A}), \mu^+_\pi(\mathbb{A})]_\sqsubset$ are π-closed. If π satisfies proof conditions (1) and (2), then $\mathcal{G}$ cannot be π-closed if for given $p \in \mathbf{P}$ there are odd number of $\mathbb{A} \in \mathcal{G}$ such that $t(\mathbb{A}) = p$.

Lemma 1. *If π satifies proof condition (4), then $\leq_\psi$ is a partial order on $\mathcal{P}_\mathbb{W}$.*

Proof. Reflexivity and transitivity directly follow from the definition of $\leq_\psi$.

Now lets prove antisymmetry. Suppose that there are $\mathbb{B}, \mathbb{C} \in \mathcal{P}_\mathbb{W}$ such that $\mathbb{B} \leq_\psi \mathbb{C}$ and $\mathbb{C} \leq_\psi \mathbb{B}$. If $\mathbb{B} \neq \mathbb{C}$ then there is $i > 0$ such that $\psi^i(\mathbb{B}) = \mathbb{B}$ and thus for all $j > 0$, $\psi^j(\mathbb{B})$ is defined.

If π satisfy proof condition (4) then if $\mathbb{A} \leq_\varphi \mathbb{B}$ then $\mathbb{A} \leq_\psi \mathbb{B}$. There is $\mathbb{A}_0 \in \mathcal{P}_\mathbb{W}$ such that $d(\mathbb{A}_0) = 0$, and for all $\mathbb{A} \in \mathcal{P}_\mathbb{W}$, $\mathbb{A} \leq_\varphi \mathbb{A}_0$. This means that $\mathbb{B} \leq_\varphi \mathbb{A}_0$ and thus $\mathbb{B} \leq_\psi \mathbb{A}_0$. The function ψ is not defined on $\mathbb{A}_0$. Contradiction.

Lemma 2. $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to B$ *if and only if there exists π satisfying proof conditions (1)-(4).*

$\mathrm{L}(\backslash, /) \vdash A_1 \dots A_n \to B$ *if and only if $n > 0$ and there exists π satisfying proof conditions (1)-(5).*

Proof. Suppose that $\mathrm{L}^{(*)}(\backslash, /) \vdash A_1 \dots A_n \to B$. Induction on the length of the derivation.

If the sequent is of the form $p \to p$, then $\mathbb{W} = p^1 p^0$, $\mathcal{P}_{\mathbb{W}} = \{p^1, p^1 p^0\}$, $\mathcal{N}_{\mathbb{W}} = \{p^1\}$ and π such that $\pi(p^1) = p^1 p^0$ and $\pi(p^1 p^0) = p^1$ satisfies all necessary proof conditions.

Suppose that the last step in the derivation of $A_1 \dots A_n \to B$ was an application of the rule $(\to /)$. Then $B = (C/D)$, $\mathrm{L}^{(*)}(\backslash, /) \vdash A_1 \dots A_n D \to C$ and for $\mathcal{P}_{\mathbb{W}'}$, where $\mathbb{W}' = [\![A_1]\!]^{\to} \dots [\![A_n]\!]^{\to} [\![D]\!]^{\to} [\![C]\!]$ there exists π' satisfying all necessary proof conditions. But in this case $\mathbb{W} = \mathbb{W}'$, and therefore this π' works for the sequent $A_1 \dots A_n \to B$ too.

Suppose that the last step in the derivation of $A_1 \dots A_n \to B$ was an application of the rule $(\to \backslash)$. Then $B = (C\backslash D)$, $\mathbb{W} = [\![A_1]\!]^{\to} \dots [\![A_n]\!]^{\to} [\![D]\!] [\![C]\!]^{\leftarrow}$, $\mathrm{L}^{(*)}(\backslash, /) \vdash C A_1 \dots A_n \to D$, and by induction hypothesis for $\mathcal{P}_{\mathbb{W}'}$, where

$$\mathbb{W}' = [\![C]\!]^{\to} [\![A_1]\!]^{\to} \dots [\![A_n]\!]^{\to} [\![D]\!]$$

there exists π' satisfying all necessary proof conditions. Consider

$$\beta : \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta(\mathbb{A}) = \begin{cases} [\![A_1]\!]^{\to} \dots [\![A_n]\!]^{\to} [\![D]\!] (\mathbb{A}^{\to^{-1}})^{\leftarrow}, & \text{if } \mathbb{A} \sqsubseteq [\![C]\!]^{\to}; \\ [\![C]\!]^{\to} \diagdown \mathbb{A}, & \text{if } [\![C]\!]^{\to} \sqsubset \mathbb{A}. \end{cases}$$

Let $\pi(\mathbb{A}) = \beta(\pi'(\beta^{-1}(\mathbb{A})))$. Such π satisfies all necessary proof conditions.

Suppose that the last step in the derivation of $A_1 \dots A_n \to B$ was an application of the rule $(/ \to)$. Then $A_1 \dots A_n \to B$ is of the form

$$C_1 \dots (C_i/D) D_1 \dots D_k C_{i+1} \dots C_l \to C$$

so that $\mathrm{L}^{(*)}(\backslash, /) \vdash C_1 \dots C_l \to C$ and $\mathrm{L}^{(*)}(\backslash, /) \vdash D_1 \dots D_k \to D$.

Consider $\mathbb{W}' = [\![C_1]\!]^{\to} \dots [\![C_l]\!]^{\to} [\![C]\!]$ and $\mathbb{W}'' = [\![D_1]\!]^{\to} \dots [\![D_k]\!]^{\to} [\![D]\!]$. By induction hypothesis there are π' and π'' — functions on $\mathcal{P}_{\mathbb{W}'}$ and $\mathcal{P}_{\mathbb{W}''}$ respectively, satisfying all necessary proof conditions.

Let $\mathbb{C} = [\![C_1]\!]^{\to} \dots [\![C_i]\!]^{\to}$ and $\mathbb{D} = [\![D_1]\!]^{\to} \dots [\![D_k]\!]^{\to}$. Consider

$$\beta' \colon \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta'(\mathbb{A}) = \begin{cases} \mathbb{A} & , \text{if } \mathbb{A} \sqsubseteq \mathbb{C}; \\ \mathbb{C}([\![D]\!]^{\to})^{\to} \mathbb{D} (\mathbb{C} \diagdown \mathbb{A}) & , \text{if } \mathbb{C} \sqsubset \mathbb{A}; \end{cases}$$

$$\text{and } \beta'' \colon \mathcal{P}_{\mathbb{W}''} \to \mathcal{P}_{\mathbb{W}}, \beta''(\mathbb{A}) = \begin{cases} \mathbb{C}([\![D]\!]^{\to})^{\to} \mathbb{A} & , \text{if } \mathbb{A} \sqsubseteq \mathbb{D}; \\ \mathbb{C}((\mathbb{D} \diagdown \mathbb{A})^{\to})^{\to} & , \text{if } \mathbb{D} \sqsubset \mathbb{A}; \end{cases}.$$

$$\text{Let } \pi(\mathbb{A}) = \begin{cases} \beta'(\pi'(\beta'^{-1}(\mathbb{A}))) & , \text{if } \mathbb{A} \sqsubseteq \mathbb{C} \text{ or } \mathbb{C}([\![D]\!]^{\to})^{\to} \mathbb{D} \sqsubset \mathbb{A}; \\ \beta''(\pi''(\beta''^{-1}(\mathbb{A}))) & , \text{if } \mathbb{C} \sqsubset \mathbb{A} \sqsubseteq \mathbb{C}([\![D]\!]^{\to})^{\to} \mathbb{D}; \end{cases}.$$

Such π satisfies all necessary proof conditions.

Suppose that the last step in the derivation of $A_1 \dots A_n \to B$ was an application of the rule $(\backslash \to)$. Then $A_1 \dots A_n \to B$ is of the form

$$C_1 \dots C_{i-1} D_1 \dots D_k (D\backslash C_i) \dots C_l \to C$$

so that $\mathrm{L}^{(*)}(\backslash, /) \vdash C_1 \dots C_l \to C$ and $\mathrm{L}^{(*)}(\backslash, /) \vdash D_1 \dots D_k \to D$.

Consider $\mathbb{W}' = [\![C_1]\!]^{\rightarrow} \dots [\![C_l]\!]^{\rightarrow}[\![C]\!]$ and $\mathbb{W}'' = [\![D_1]\!]^{\rightarrow} \dots [\![D_k]\!]^{\rightarrow}[\![D]\!]$. By induction hypothesis there are π' and π'' — functions on $\mathcal{P}_{\mathbb{W}'}$ and $\mathcal{P}_{\mathbb{W}''}$ respectively, satisfying all necessary proof conditions.

Let $\mathbb{C} = [\![C_1]\!]^{\rightarrow} \dots [\![C_{i-1}]\!]^{\rightarrow}$ and $\mathbb{D} = [\![D_1]\!]^{\rightarrow} \dots [\![D_k]\!]^{\rightarrow}$. Consider

$$\beta' \colon \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta'(\mathbb{A}) = \begin{cases} \mathbb{A} & \text{, if } \mathbb{A} \sqsubseteq \mathbb{C}; \\ \mathbb{C}\mathbb{D}([\![D]\!]^{\leftarrow})^{\rightarrow}(\mathbb{C} \diagdown \mathbb{A}) & \text{, if } \mathbb{C} \sqsubset \mathbb{A}; \end{cases}$$

$$\text{and } \beta'' \colon \mathcal{P}_{\mathbb{W}''} \to \mathcal{P}_{\mathbb{W}}, \beta''(\mathbb{A}) = \begin{cases} \mathbb{C}\mathbb{A} & \text{, if } \mathbb{A} \sqsubseteq \mathbb{D}; \\ \mathbb{C}\mathbb{D}((\mathbb{D} \diagdown \mathbb{A})^{\leftarrow})^{\rightarrow} & \text{, if } \mathbb{D} \sqsubset \mathbb{A}; \end{cases}.$$

$$\text{Let } \pi(\mathbb{A}) = \begin{cases} \beta'(\pi'(\beta'^{-1}(\mathbb{A}))) & \text{, if } \mathbb{A} \sqsubseteq \mathbb{C} \text{ or } \mathbb{C}\mathbb{D}([\![D]\!]^{\leftarrow})^{\rightarrow} \sqsubset \mathbb{A}; \\ \beta''(\pi''(\beta''^{-1}(\mathbb{A}))) & \text{, if } \mathbb{C} \sqsubset \mathbb{A} \sqsubseteq \mathbb{C}\mathbb{D}([\![D]\!]^{\leftarrow})^{\rightarrow}; \end{cases}.$$

Such π satisfies all necessary proof conditions.

Thus we proved one side of the lemma.

Now suppose that for the given sequent $A_1 \dots A_n \to B$, for $\mathcal{P}_{\mathbb{W}}$ there exists π satisfying proof conditions (1)-(4).

Induction on total number of connectives in the sequent.

If there are no connectives, the sequent is of the form $p_1 \dots p_n \to q$ and $\mathbb{W} = p_1^{(1)} \dots p_n^{(1)} q^{(0)}$. The function π satisfies proof condition (1), thus $|\mathcal{N}_{\mathbb{W}}| = |\mathcal{P}_{\mathbb{W}} \setminus \mathcal{N}_{\mathbb{W}}|$. This means that $n = 1$. So $\mathcal{P}_{\mathbb{W}} = \{p_1^{(1)}, p_1^{(1)} q^{(0)}\}$ and $\mathcal{N}_{\mathbb{W}} = \{p_1^{(1)}\}$. The function π satisfies proof condition (2), therefore $p_1 = q$, and the sequent is an axiom.

If $B = (C/D)$, then the sequent $A_1 \dots A_n D \to C$ has less connectives then the original sequent, but $[\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow}[\![D]\!]^{\rightarrow}[\![C]\!] = \mathbb{W}$, and therefore π satisfies all necessary proof conditions for the new sequent. By inductional hypothesis this means that $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n D \to C$ and by applying the rule $(\to /)$ we get $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to B$.

If $B = (C \backslash D)$, then the sequent $C A_1 \dots A_n \to D$ has less connectives then the original sequent.

Let $\mathbb{W}' = [\![C]\!]^{\rightarrow}[\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow}[\![D]\!]$. Consider

$$\beta : \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta(\mathbb{A}) = \begin{cases} [\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow}[\![D]\!](\mathbb{A}^{\rightarrow^{-1}})^{\leftarrow} & \text{, if } \mathbb{A} \sqsubseteq [\![C]\!]^{\rightarrow}; \\ [\![C]\!]^{\rightarrow} \diagdown \mathbb{A} & \text{, if } [\![C]\!]^{\rightarrow} \sqsubset \mathbb{A}; \end{cases}.$$

Let $\pi'(\mathbb{A}) = \beta^{-1}(\pi(\beta(\mathbb{A})))$. Such π' satisfies all necessary proof conditions. By induction hypothesis this means that $\mathrm{L}^*(\backslash, /) \vdash C A_1 \dots A_n \to D$, and by applying the rule $(\to \backslash)$ we get $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to B$.

Now we can only consider sequents of the form $A_1 \dots A_n \to p$. This means theat $\mathbb{W} = [\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow} p^{(0)}$. Let $\mathbb{A}_1 = \pi(\mathbb{W})$. Since π satisfies proof condition (4) and ψ is not defined on $\mathbb{W}$, $\varphi(\mathbb{A}_1) = \mathbb{W}$. Therefore $d(\mathbb{A}_1) = 1$. Let $\mathbb{A}_1 = [\![A_1]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow}\mathbb{A}''$.

Suppose that $A_i = (C/D)$. This means that $[\![A_i]\!]^{\rightarrow} = [\![C]\!]^{\rightarrow}([\![D]\!]^{\rightarrow})^{\rightarrow}$. There exists a unique $\mathbb{D}_0 \in \mathcal{P}_{[\![D]\!]}$ such that $d(\mathbb{D}_0) = 0$. Consider

$$\mathbb{A}_2 = [\![A_1]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow} [\![C]\!]^{\rightarrow} (\mathbb{D}_0^{\rightarrow})^{\rightarrow} \in \mathcal{P}_{\mathbb{W}}.$$

$d(\mathbb{A}_2) = -2$, $\varphi(\mathbb{A}_2) = \mathbb{A}_1$, $\psi^2(\mathbb{A}_2) = \mathbb{W}$, and there is no $\mathbb{B} \in \mathcal{P}_{\mathbb{W}}$ such that $\mathbb{A}_2 \sqsubset \mathbb{B}$ and $\varphi(\mathbb{B}) = \mathbb{A}_1$.

Also $\mathcal{F}_{\psi}(\mathbb{A}_2) = [\nu_{\psi}^{-}(\mathbb{A}_2), \nu_{\psi}^{+}(\mathbb{A}_2)]_{\sqsubset}$. Let us prove this statement. Consider $\mathbb{B} \in [\nu_{\psi}^{-}(\mathbb{A}), \nu_{\psi}^{+}(\mathbb{A})]_{\sqsubset}, \mathbb{B} \neq \mathbb{A}_2$. There exists $\mathbb{C} \in \mathcal{F}_{\psi}(\mathbb{A}_2)$ such that $\mathbb{B} \sqsubseteq \mathbb{C}$ and $\psi(\mathbb{C}) \sqsubseteq \mathbb{B}$. If $\mathbb{C} \in \mathcal{P}_{\mathbb{W}} \setminus \mathcal{N}_{\mathbb{W}}$, then $\mathbb{B} \in [\mu_{\varphi}^{-}(\mathbb{C}), \mu_{\varphi}^{+}(\mathbb{C})]_{\sqsubset}$, and thus $\mathbb{B} \leq_{\psi} \varphi(\mathbb{C}) \leq_{\psi} \mathbb{A}_2$. If $\mathbb{C} \in \mathcal{P}_{\mathbb{W}}$, then $\pi(\mathbb{B}) \in [\mu_{\varphi}^{-}(\mathbb{C}), \mu_{\varphi}^{+}(\mathbb{C})]_{\sqsubset} \subset [\nu_{\psi}^{-}(\mathbb{A}), \nu_{\psi}^{+}(\mathbb{A})]_{\sqsubset}$. Since $\mathbb{B} <_{\psi} \mathbb{A}_1$, this means that $\mathbb{B} <_{\psi} \mathbb{A}_2$.

Let $\nu_{\psi}^{+}(\mathbb{A}_2) = [\![A_1]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow}$. Consider

$$\mathbb{W}' = [\![A_1]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow} [\![C]\!]^{\rightarrow} [\![A_{l+1}]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow} p^{(0)}$$

and $\mathbb{W}'' = [\![A_{i+1}]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow} [\![D]\!]$. Let $\mathbb{C} = [\![A_1]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow} [\![C]\!]^{\rightarrow}$ and $\mathbb{D} = [\![A_{i+1}]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow}$. Consider

$$\beta' \colon \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta'(\mathbb{A}) = \begin{cases} \mathbb{A} & , \text{if } \mathbb{A} \sqsubseteq \mathbb{C}; \\ \mathbb{C}([\![D]\!]^{\rightarrow})^{\rightarrow} \mathbb{D}(\mathbb{C} \diagdown \mathbb{A}) & , \text{if } \mathbb{C} \sqsubset \mathbb{A}; \end{cases},$$

$$\text{and } \beta'' \colon \mathcal{P}_{\mathbb{W}''} \to \mathcal{P}_{\mathbb{W}}, \beta''(\mathbb{A}) = \begin{cases} \mathbb{C}([\![D]\!]^{\rightarrow})^{\rightarrow} \mathbb{A} & , \text{if } \mathbb{A} \sqsubseteq \mathbb{D}; \\ \mathbb{C}((\mathbb{D} \diagdown \mathbb{A})^{\rightarrow})^{\rightarrow} & , \text{if } \mathbb{D} \sqsubset \mathbb{A}; \end{cases}.$$

Functions $\pi' = \beta'^{-1} \pi \beta'$ and $\pi'' = \beta''^{-1} \pi \beta''$ satisfy all necessary proof conditions. By induction hypothesis this means that

$$\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_{i-1} C A_{l+1} \dots A_n \to p$$

and $\mathrm{L}^*(\backslash, /) \vdash A_{i+1} \dots A_l \to D$. By applying the rule $(/ \to)$ we get

$$\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to p.$$

Suppose that $A_i = (D \backslash C)$. This means that $[\![A_i]\!]^{\rightarrow} = ([\![D]\!]^{\leftarrow})^{\rightarrow} [\![C]\!]^{\rightarrow}$. There exists a unique $\mathbb{D}_0 \in \mathcal{P}_{[\![D]\!]}$ such that $d(\mathbb{D}_0) = 0$. Consider

$$\mathbb{A}_2 = [\![A_1]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow} (\mathbb{D}_0^{\leftarrow})^{\rightarrow} \in \mathcal{P}_{\mathbb{W}}.$$

$d(\mathbb{A}_2) = 2$, $\varphi(\mathbb{A}_2) = \mathbb{A}_1$, $\psi^2(\mathbb{A}_2) = \mathbb{W}$, and there is no $\mathbb{B} \in \mathcal{P}_{\mathbb{W}}$ such that $\mathbb{B} \sqsubset \mathbb{A}_2$ and $\varphi(\mathbb{B}) = \mathbb{A}_1$. Like in previous case we can say that $\mathcal{F}_{\psi}(\mathbb{A}_2) = [\nu_{\psi}^{-}(\mathbb{A}_2), \nu_{\psi}^{+}(\mathbb{A}_2)]_{\sqsubset}$.

Let $\nu_{\psi}^{-}(\mathbb{A}_2) = [\![A_1]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow} q^{(j)}$ for some $q^{(j)} \in \mathrm{At}$. Consider

$$\mathbb{W}' = [\![A_1]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow} [\![C]\!]^{\rightarrow} [\![A_{i+1}]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow} p^{(0)}$$

and $\mathbb{W}'' = [\![A_{l+1}]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow} [\![D]\!]$.

Let $\mathbb{C} = [\![A_1]\!]^{\rightarrow} \dots [\![A_l]\!]^{\rightarrow}$ and $\mathbb{D} = [\![A_{l+1}]\!]^{\rightarrow} \dots [\![A_{i-1}]\!]^{\rightarrow}$. Consider

$$\beta' \colon \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}, \beta'(\mathbb{A}) = \begin{cases} \mathbb{A} & \text{, if } \mathbb{A} \sqsubseteq \mathbb{C}; \\ \mathbb{C}\mathbb{D}([\![D]\!]^{\leftarrow})^{\rightarrow}(\mathbb{C} \diagdown \mathbb{A}) & \text{, if } \mathbb{C} \sqsubset \mathbb{A}; \end{cases},$$

$$\text{and } \beta'' \colon \mathcal{P}_{\mathbb{W}''} \to \mathcal{P}_{\mathbb{W}}, \beta''(\mathbb{A}) = \begin{cases} \mathbb{C}\mathbb{A} & \text{, if } \mathbb{A} \sqsubseteq \mathbb{D}; \\ \mathbb{C}\mathbb{D}((\mathbb{D} \diagdown \mathbb{A})^{\leftarrow})^{\rightarrow} & \text{, if } \mathbb{D} \sqsubset \mathbb{A}; \end{cases}.$$

Functions $\pi' = \beta'^{-1}\pi\beta'$ and $\pi'' = \beta''^{-1}\pi\beta''$ satisfy all necessary proof conditions. By induction hypothesis this means that

$$\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_l C A_{i+1} \dots A_n \to p$$

and $\mathrm{L}^*(\backslash, /) \vdash A_{l+1} \dots A_{i-1} \to D$. By applying the rule $(\backslash \to)$ we get

$$\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to p.$$

Thus we fully proved the lemma for $\mathrm{L}^*(\backslash, /)$.

Suppose we have π that satisfies proof conditions (1)-(5). We already proved that $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to B$. The construction given provides us with possible last step of the derivation. Hence we can construct a derivation. If π satisfies proof condition (5) this means that there will be no $\mathbb{A}_2$ such that $\mathcal{F}_\psi(\mathbb{A}_2) = \mathcal{F}_\varphi(\mathbb{A}_2)$, and thus there will be no steps in derivation that require sequents of the form $\to A$. Thus $\mathrm{L}(\backslash, /) \vdash A_1 \dots A_n \to B$.

The lemma is fully proven.

Lemma 3. *Suppose we have two sequents $A_1 \dots A_n \to B$ and $C_1 \dots C_m \to D$, and $\mathrm{L}^*(\backslash, /) \vdash A_1 \dots A_n \to B$.*

Let $\mathbb{W} = [\![A_1]\!]^{\rightarrow} \dots [\![A_n]\!]^{\rightarrow}[\![B]\!]$ and $\mathbb{W}' = [\![C_1]\!]^{\rightarrow} \dots [\![C_m]\!]^{\rightarrow}[\![D]\!]$. Suppose that there is a mapping $\beta \colon \mathcal{P}_{\mathbb{W}'} \to \mathcal{P}_{\mathbb{W}}$ such that the following holds:

1. *β is injective,*
2. *For all $\mathbb{A} \in \mathcal{P}_{\mathbb{W}'}$, $t(\beta(\mathbb{A})) = t(\mathbb{A})$, $d(\beta(\mathbb{A})) = d(\mathbb{A})$,*
3. *For all $\mathbb{A}, \mathbb{B} \in \mathbb{P}_{\mathbb{W}'}$, $\mathbb{A} \sqsubset \mathbb{B}$ if and only if $\beta(\mathbb{A}) \sqsubset \beta(\mathbb{B})$.*

Let $\mathcal{G} = \{\mathbb{A} \in \mathcal{P}_{\mathbb{W}} \mid \neg\exists \mathbb{B} \in \mathcal{P}_{\mathbb{W}'}, \beta(\mathbb{B}) = \mathbb{A}\}$. If $\mathcal{G}$ is π-closed and φ-closed, then $\mathrm{L}^*(\backslash, /) \vdash C_1 \dots C_n \to D$.

Proof. Let φ' be φ for $\mathcal{P}_{\mathbb{W}'}$. Since $\mathcal{G}$ is φ-closed, for all $\mathbb{A} \in \mathcal{P}_{\mathbb{W}'}$, $\varphi'(\mathbb{A}) = \beta^{-1}(\varphi(\beta(\mathbb{A})))$. Since $\mathcal{G}$ is π-closed, π' defined as $\beta^{-1}\pi\beta$ is defined on all $\mathcal{P}_{\mathbb{W}'}$ and satisfies proof conditions (1)-(4). Therefore by lemma 2

$$\mathrm{L}^*(\backslash, /) \vdash C_1 \dots C_n \to D.$$

Lemma 4. *Suppose $\mathbb{A} \in \mathcal{N}_{\mathbb{W}}$ and $\mathbb{B}$ is the φ-join of $\mathbb{A}$ and $\pi(\mathbb{A})$. If π satisfies proof conditions (1)-(4), then $\mathbb{B} \notin \mathcal{N}_{\mathbb{W}}$.*

Proof. Suppose that $\mathbb{B}_i \in \mathcal{N}_{\mathbb{W}}$. There is $\mathbb{B}_1$ such that $\mathbb{A} <_\psi \mathbb{B}_1$ and $\varphi(\mathbb{B}_1) = \mathbb{B}$. There is $\mathbb{B}_2$ such that $\pi(\mathbb{A}) <_\psi \mathbb{B}_2$ and $\varphi(\mathbb{B}_2) = \mathbb{B}$. This means that $\mathbb{A} \leq_\psi \mathbb{B}_1$ and $\pi(\mathbb{A}) \leq_\psi \mathbb{B}_2$ and since $\psi(\mathbb{A}) = \pi(\mathbb{A})$, either $\mathbb{B}_1 <_\psi \mathbb{B}_2$ or $\mathbb{B}_2 <_\psi \mathbb{B}_1$. But since $\psi(\mathbb{B}_1) = \psi(\mathbb{B}_2) = \mathbb{B}$, we get $\mathbb{B} <_\psi \mathbb{B}$. Contradiction.

4 Proof of the Main Theorem

Consider $\mathbb{W} = [\![F_1(t_1)]\!]^{\rightarrow} \dots [\![F_n(t_n)]\!]^{\rightarrow}[\![G]\!]$.

If a primitive type occurs in the sequent $F_1(t_1)\dots F_n(t_n) \rightarrow G$ it occurs exactly twice. Let $\mathbb{P}_i^{j+}$ be the element of $\mathcal{N}_{\mathbb{W}}$ such that $t(\mathbb{P}_i^{j+}) = p_i^j$ and $\mathbb{P}_i^{j-}$ be the element of $\mathcal{P}_{\mathbb{W}} \setminus \mathcal{N}_{\mathbb{W}}$ such that $t(\mathbb{P}_i^{j-}) = p_i^j$. In the same way we define $\mathbb{Q}_i^{j+}$ and $\mathbb{Q}_i^{j-}$.

The function π can only satisfy proof conditions (1) and (2) if for every i and j, $\pi(\mathbb{P}_i^{j+}) = \mathbb{P}_i^{j-}$ and $\pi(\mathbb{Q}_i^{j+}) = \mathbb{Q}_i^{j-}$. If it is so then π satisfies proof conditions (3) and (5).

The following facts hold:

1. $d(P_i^{m-}) = 0$.
2. If $\neg_{t_i} x_i$ does not appear in clause c_j, then $\varphi^3(\mathbb{P}_i^{j-1+}) = \varphi^2(\mathbb{Q}_i^{j-}) = \varphi(\mathbb{Q}_{i-1}^{j+}) = \mathbb{P}_{i-1}^{j-}$.
3. If $\neg_{t_i} x_i$ appears in clause c_j, then $\varphi^3(\mathbb{Q}_{i-1}^{j+}) = \varphi^2(\mathbb{P}_{i-1}^{j-}) = \varphi(\mathbb{P}_i^{j-1+}) = \mathbb{Q}_i^{j-}$.
4. $\varphi^4(\mathbb{Q}_0^{j-}) = \varphi^3(\mathbb{P}_0^{j+}) = \varphi^2(\mathbb{P}_n^{j-1-}) = \varphi(\mathbb{Q}_n^{j+}) = \mathbb{P}_n^{j-}$.

Lemma 5. *For every $0 < i \leq n$ and $j > 0$, $\mathbb{P}_i^{j-1+} <_\psi \mathbb{Q}_i^{j-}$.*

Proof. For $i = n$ this is true, because

$$\psi^3(\mathbb{P}_n^{j-1+}) = \pi\varphi\pi(\mathbb{P}_n^{j-1+}) = \pi\varphi(\mathbb{P}_n^{j-1-}) = \pi(\mathbb{Q}_n^{j+}) = \mathbb{Q}_n^{j-}.$$

Now suppose that for all $i' > i$ this was already proven. There are four possibilities:

1. If $\neg_{t_{i+1}} x_{i+1}$ does not appear in clauses c_{j-1} and c_j, then $\psi^2(\mathbb{P}_i^{j-1+}) = \mathbb{P}_{i+1}^{j-1+}$, $\psi^2(\mathbb{Q}_{i+1}^{j-}) = \mathbb{Q}_i^{j-}$, and $\mathbb{P}_{i+1}^{j-1+} <_\psi \mathbb{Q}_{i+1}^{j-}$. Thus $\mathbb{P}_i^{j-1+} <_\psi \mathbb{Q}_i^{j-}$.
2. If $\neg_{t_{i+1}} x_{i+1}$ does not appear in the clause c_{j-1}, but appears in c_j, then $\psi^3(\mathbb{P}_i^{j-1+}) = \pi\varphi\pi(\mathbb{P}_i^{j-1+}) = \pi\varphi(\mathbb{P}_i^{j-1-}) = \pi(\mathbb{Q}_i^{j+}) = \mathbb{Q}_i^{j-}$.
3. If $\neg_{t_{i+1}} x_{i+1}$ appears in the clause c_{j-1}, but does not appear in c_j, then $\psi^2(\mathbb{P}_i^{j-1+}) = \mathbb{P}_{i+1}^{j-2+}$, $\psi^2(\mathbb{Q}_{i+1}^{j}) = \mathbb{Q}_i^{j-}$, $\varphi(\mathbb{Q}_{i+1}^{j-1+}) = \mathbb{P}_{i+1}^{j-1+}$, $\mathbb{P}_{i+1}^{j-2+} <_\psi \mathbb{Q}_{i+1}^{j-1-}$, and $\mathbb{P}_{i+1}^{j-1+} <_\psi \mathbb{Q}_{i+1}^{j-}$. Thus $\mathbb{P}_i^{j-1+} <_\psi \mathbb{Q}_i^{j-}$.
4. If $\neg_{t_{i+1}} x_{i+1}$ appears in both clauses c_{j-1} and c_j, then $\psi^2(\mathbb{P}_i^{j-1+}) = \mathbb{P}_{i+1}^{j-2+}$, $\psi^2(\mathbb{Q}_{i+1}^{j-1-}) = \mathbb{Q}_i^{j-}$, and $\mathbb{P}_{i+1}^{j-2+} <_\psi \mathbb{Q}_{i+1}^{j-1-}$. Thus $\mathbb{P}_i^{j-1+} <_\psi \mathbb{Q}_i^{j-}$.

Lemma 6. *For every $0 \leq i < n$ and $j > 0$, $\mathbb{Q}_i^{j+} <_\psi \mathbb{P}_i^{j-}$.*

Proof. For $i = 0$ this is true, because

$$\psi^3(\mathbb{Q}_0^{j+}) = \pi\varphi\pi(\mathbb{Q}_0^{j+}) = \pi\varphi(\mathbb{Q}_0^{j-}) = \pi(\mathbb{P}_0^{j+}) = \mathbb{P}_0^{j-}.$$

Now suppose that for all $i' < i$ this was already proven. There are four possibilities:

1. If $\neg_{t_i} x_i$ does not appear in clauses c_{j+1} and c_j, then $\psi^2(\mathbb{Q}_i^{j+}) = \mathbb{Q}_{i-1}^{j+}$, $\psi^2(\mathbb{P}_{i-1}^{j-}) = \mathbb{P}_i^{j-}$, and $\mathbb{Q}_{i-1}^{j+} <_\psi \mathbb{P}_{i-1}^{j-}$. Thus $\mathbb{Q}_i^{j+} <_\psi \mathbb{P}_i^{j-}$.

2. If $\neg_{t_i} x_i$ does not appear in the clause c_{j+1}, but appears in c_j, then $\psi^3(\mathbb{Q}_i^{j+}) = \pi\varphi\pi(\mathbb{Q}_i^{j+}) = \pi\varphi(\mathbb{Q}_i^{j-}) = \pi(\mathbb{P}_i^{j+}) = \mathbb{P}_i^{j-}$.
3. If $\neg_{t_i} x_i$ appears in the clause c_{j+1}, but does not appear in c_j, then $\psi^2(\mathbb{Q}_i^{j+}) = \mathbb{Q}_{i-1}^{j+}$, $\psi^2(\mathbb{P}_{i-1}^{j+1-}) = \mathbb{P}_i^{j-}$, $\varphi(\mathbb{P}_{i-1}^{j+}) = \mathbb{Q}_{i-1}^{j+1+}$, $\mathbb{Q}_{i-1}^{j+} <_\psi \mathbb{P}_{i-1}^{j-}$, and $\mathbb{Q}_{i-1}^{j+1+} <_\psi \mathbb{P}_{i-1}^{j+1-}$. Thus $\mathbb{Q}_i^{j+} <_\psi \mathbb{P}_i^{j-}$.
4. If $\neg_{t_i} x_i$ appears in both clauses c_{j+1} and c_j, then $\psi^2(\mathbb{Q}_i^{j+}) = \mathbb{Q}_{i-1}^{j+1+}$, $\psi^2(\mathbb{P}_{i-1}^{j+1-}) = \mathbb{P}_i^{j-}$, and $\mathbb{Q}_{i-1}^{j+1+} <_\psi \mathbb{P}_{i-1}^{j+1-}$. Thus $\mathbb{Q}_i^{j+} <_\psi \mathbb{P}_i^{j-}$.

From lemmas 5 and 6 we can conclude that if $i > 0$ and $j \leq j'$ then $\mathbb{P}_i^{j+} <_\psi \mathbb{P}_i^{j'+}$.

Lemma 7. *If $i < i'$, then $\mathbb{P}_i^{j+} <_\psi \mathbb{P}_{i'}^{j+}$.*

Proof. If $\neg_{t_{i+1}} x_{i+1}$ appears in clause c_j, then $\psi^2(\mathbb{P}_i^{j+}) = \mathbb{P}_{i+1}^{j-1+}$ and $\mathbb{P}_{i+1}^{j-1+} <_\psi \mathbb{P}_{i+1}^{j+}$. If $\neg_{t_{i+1}} x_{i+1}$ appears in clause c_{j+1}, then $\psi(\mathbb{P}_i^{j+1-}) = \mathbb{P}_{i+1}^{j+}$ and $\mathbb{P}_i^{j-} <_\psi \mathbb{P}_i^{j+1+}$. If neither of this is the case, then $\psi^2(\mathbb{P}_i^{j+}) = \mathbb{P}_{i+1}^{j+}$. This means that $\mathbb{P}_i^{j+} <_\psi \mathbb{P}_{i+1}^{j+}$ and thus $\mathbb{P}_i^{j+} <_\psi \mathbb{P}_{i'}^{j+}$.

Lemma 8. *$\langle t_1, \ldots, t_n \rangle$ is a satisfying assignment for $c_1 \wedge \ldots \wedge c_m$ if and only if $\mathrm{L}^*(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G$ and if and only if*

$$\mathrm{L}(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G.$$

Proof. Suppose that $\langle t_1, \ldots, t_n \rangle$ is a satisfying assignment for $c_1 \wedge \ldots \wedge c_m$. In view of lemmas 5 and 6 the only members of $\mathcal{N}_{\mathbb{W}}$ for which we have not proved that π satisfies proof condition (4) are $\mathbb{P}_0^{j+}$.

We now prove that for every j, $\mathbb{P}_0^{j+} <_\psi \varphi(\mathbb{P}_0^{j+}) = \mathbb{P}_n^{j-1-}$. There exist i such that $\neg_{t_i} x_i$ appears in clause c_j. This means that $\psi(\mathbb{P}_{i-1}^{j-}) = \mathbb{P}_i^{j-1+}$ and by lemma 7 $\mathbb{P}_0^{j+} <_\psi \mathbb{P}_i^{j+}$ and $\mathbb{P}_i^{j-1+} <_\psi \mathbb{P}_n^{j-1+}$. Thus $\mathbb{P}_0^{j+} <_\psi \varphi(\mathbb{P}_0^{j+}) = \mathbb{P}_n^{j-1-}$ and by lemma 2 we can now say that $\mathrm{L}^*(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G$.

Suppose that $\langle t_1, \ldots, t_n \rangle$ is not a satisfying assignment for $c_1 \wedge \ldots \wedge c_m$. There exists j such that no $\neg_{t_i} x_i$ appear in clause c_j. This means that for $i \leq n$, $\psi^{2i}(\mathbb{Q}_n^{j+}) = \mathbb{Q}_{n-i}^{j+}$, $\psi(\mathbb{P}_n^{j-1-}) = \mathbb{Q}_n^{j+}$, and $\psi(\mathbb{Q}_0^{j-}) = \mathbb{P}_0^{j+}$. Thus $\mathbb{P}_n^{j-1-} <_\psi \mathbb{P}_0^{j+}$. This means that π cannot satisfy proof condition (4). Thus by lemma 1 we have $\mathrm{L}^*(\backslash, /) \nvdash F_1(t_1) \ldots F_n(t_n) \to G$.

Since π satisfies proof condition (5),

$$\mathrm{L}(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G \Leftrightarrow \mathrm{L}^*(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G$$

and thus the lemma is fully proven.

Lemma 9. *If $\mathrm{L}(\backslash, /) \vdash \Pi \to A$ and $\Pi' \to A'$ is the result of changing all instances of primitive type p to primitive type q, then $\mathrm{L}(\backslash, /) \vdash \Pi' \to A'$.*

Proof. If we change p to q throughout the derivation of $\Pi \to A$, we will get the derivation of $\Pi' \to A'$.

Lemma 10.

$$\begin{aligned}
&\mathrm{L}(\backslash, /) \vdash H_i \to B_i \backslash A_i,\\
&\mathrm{L}(\backslash, /) \vdash H_i \to D_i \backslash C_i,\\
&\mathrm{L}(\backslash, /) \vdash F_i(1) \to B \backslash A,\\
&\mathrm{L}(\backslash, /) \vdash F_i(0) \to D \backslash C.
\end{aligned}$$

Proof. Consider boolean formula $\underbrace{(x_1 \vee x_2) \wedge x_1 \wedge \ldots \wedge x_1}_{m \text{ clauses}}$.

Let $F_1'(1)F_2'(0) \to G'$ be the sequent constructed for this formula. By Lemma 8 we can say that $\mathrm{L}(\backslash, /) \vdash F_1'(1)F_2'(0) \to G'$.

By changing p_0^j to a_i^j, q_0^j to b_i^j, p_1^j to p_{i-1}^j, q_1^j to q_{i-1}^j, p_2^j to p_i^j, and q_2^j to q_i^j, we get $B_iH_i \to A_i$. By Lemma 9 this means that $\mathrm{L}(\backslash, /) \vdash B_iH_i \to A_i$.

Therefore $\mathrm{L}(\backslash, /) \vdash H_i \to B_i \backslash A_i$.

By changing p_0^j to c_i^j, q_0^j to d_i^j, p_1^j to p_{i-1}^j, q_1^j to q_{i-1}^j, p_2^j to p_i^j, and q_2^j to q_i^j, we get $D_iH_i \to C_i$. By Lemma 9 this means that $\mathrm{L}(\backslash, /) \vdash D_iH_i \to C_i$.

Therefore $\mathrm{L}(\backslash, /) \vdash H_i \to D_i \backslash C_i$.

Consider boolean formula $c_1' \wedge \ldots \wedge c_m'$, where

$$c_i' = \begin{cases} (x_1 \vee x_2), & \text{if literal } \neg_1 x_i \text{ appears in } c_j \\ x_1, & \text{if literal } \neg_1 x_i \text{ doesn't appear in } c_j \end{cases}$$

Let $F_1'(1)F_2'(1) \to G'$ be the sequent constructed for this formula. By Lemma 8 we can say that $\mathrm{L}(\backslash, /) \vdash F_1'(1)F_2'(1) \to G'$.

By changing p_0^j to a_i^j, q_0^j to b_i^j, p_1^j to p_{i-1}^j, q_1^j to q_{i-1}^j, p_2^j to p_i^j, and q_2^j to q_i^j, we get $B_iF_i(1) \to A_i$. By Lemma 9 this means that $\mathrm{L}(\backslash, /) \vdash B_iF_i(1) \to A_i$.

Therefore $\mathrm{L}(\backslash, /) \vdash F_i(1) \to B_i \backslash A_i$.

Consider boolean formula $c_1' \wedge \ldots \wedge c_m'$, where

$$c_i' = \begin{cases} (x_1 \vee x_2), & \text{if literal } \neg_0 x_i \text{ appears in } c_j \\ x_1, & \text{if literal } \neg_0 x_i \text{ doesn't appear in } c_j \end{cases}$$

Let $F_1'(1)F_2'(1) \to G'$ be the sequent constructed for this formula. By Lemma 8 we can say that $\mathrm{L}(\backslash, /) \vdash F_1'(1)F_2'(1) \to G'$.

By changing p_0^j to c_i^j, q_0^j to d_i^j, p_1^j to p_{i-1}^j, q_1^j to q_{i-1}^j, p_2^j to p_i^j, and q_2^j to q_i^j, we get $D_iF_i(0) \to C_i$. By Lemma 9 this means that $\mathrm{L}(\backslash, /) \vdash D_iF_i(0) \to C_i$.

Therefore $\mathrm{L}(\backslash, /) \vdash F_i(0) \to D_i \backslash C_i$.

Lemma 11. $\mathrm{L}(\backslash, /) \vdash \Pi_i \to F_i(t_i)$, *where* $t_i \in \{0, 1\}$.

Proof. Using Lemma 10 we get

$$\dfrac{\dfrac{H_i \to D_i \backslash C_i \quad F_i(1) \to B_i \backslash A_i}{H_i(D_i \backslash C_i) \backslash F_i(1) \to B_i \backslash A_i}\,(\backslash \to) \qquad F_i(0) \to F_i(0)}{F_i(0)/(B_i \backslash A_i)H_i(D_i \backslash C_i) \backslash F_i(1) \to F_i(0)}\,(/ \to)$$

and

$$\dfrac{\dfrac{H_i \to B_i\backslash A_i \quad F_i(1) \to D_i\backslash C_i}{F_i(0)/(B_i\backslash A_i)H_i \to D_i\backslash C_i}\,(/\to) \qquad F_i(1) \to F_i(1)}{F_i(0)/(B_i\backslash A_i)H_i(D_i\backslash C_i)\backslash F_i(1) \to F_i(1)}\,(\backslash\to)$$

Thus $\mathrm{L}(\backslash, /) \vdash \Pi_i \to F_i(0)$ and $\mathrm{L}(\backslash, /) \vdash \Pi_i \to F_i(1)$.

Lemma 12. *If the formula $c_1 \wedge \ldots \wedge c_m$ is satisfiable, then* $\mathrm{L}(\backslash, /) \vdash \Pi_1 \ldots \Pi_n \to G$.

Proof. Suppose $\langle t_1, \ldots, t_n\rangle$ is a satisfying assignment for $c_1 \wedge \ldots \wedge c_m$. According to Lemma 8 $\mathrm{L}(\backslash, /) \vdash F_1(t_1) \ldots F_n(t_n) \to G$. Now we apply Lemma 11 and the cut rule n times.

Suppose $\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \ldots \Pi_n \to G$. Consider

$$\mathbb{W} = [\![F_1(0)/(B_1\backslash A_1)]\!]^{\to}[\![H_1]\!]^{\to}[\![(D_1\backslash C_1)\backslash F_1(1)]\!]^{\to} \ldots \\ \ldots [\![F_n(0)/(B_n\backslash A_n)]\!]^{\to}[\![H_n]\!]^{\to}[\![(D_n\backslash C_n)\backslash F_n(1)]\!]^{\to}[\![G]\!]$$

By Lemma 2 for $\mathcal{P}_{\mathbb{W}}$ there exists π satisfying proof conditions (1)-(4).
Consider the following abbreviations:

$$\begin{aligned}
&\mathbb{F}_i^0 = [\![F_1(0)/(B_1\backslash A_1)]\!]^{\to}[\![H_1]\!]^{\to}[\![(D_1\backslash C_1)\backslash F_1(1)]\!]^{\to} \ldots [\![F_i(0)]\!]^{\to}\\
&\mathbb{A}_i = \mathbb{F}_i^0([\![A_i]\!]^{\to})^{\to}\\
&\mathbb{B}_i = \mathbb{A}_i(([\![B_i]\!]^{\leftarrow})^{\to})^{\to}\\
&\mathbb{H}_i = \mathbb{B}_i[\![H_i]\!]^{\to}\\
&\mathbb{C}_i = \mathbb{H}_i([\![C_i]\!]^{\leftarrow})^{\to}\\
&\mathbb{D}_i = \mathbb{C}_i(([\![D_i]\!]^{\leftarrow})^{\leftarrow})^{\to}\\
&\mathbb{F}_i^1 = \mathbb{D}_i[\![F_i(1)]\!]^{\to}
\end{aligned}$$

Lemma 13. *If* $\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \ldots \Pi_i F_{i+1}(t_{i+1}) \ldots F_n(t_n) \to G$, *then there is* $t_i \in \{0, 1\}$ *such that* $\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \ldots \Pi_{i-1} F_i(t_i) \ldots F_n(t_n) \to G$

Proof. Consider $\mathbb{W}' = \mathbb{F}_i^1\mathbb{W}''$, where $\mathbb{W}'' = [\![F_{i+1}(t_{i+1})]\!]^{\to} \ldots [\![F_n(t_n)]\!]^{\to}[\![G]\!]$. By Lemma 2 for $\mathcal{P}_{\mathbb{W}'}$ there exists π satisfying proof conditions (1)-(5).

Let $\mathbb{W}'_0 = \mathbb{F}_{i-1}^1[\![F_i(0)]\!]^{\to}\mathbb{W}''$ and $\mathbb{W}'_1 = \mathbb{F}_{i-1}^1[\![F_i(1)]\!]^{\to}\mathbb{W}''$.

For each j there are only two elements of $\mathcal{P}_{\mathbb{W}'}$ such that $t(\mathbb{A}) = a_i^j$, two elements such that $t(\mathbb{A}) = b_i^j$, two elements such that $t(\mathbb{A}) = c_i^j$, and two elements such that $t(\mathbb{A}) = d_i^j$. This means that these pairs of elements are π-closed.

For each j there are six elements of $\mathcal{P}_{\mathbb{W}'}$ such that $t(\mathbb{A}) = p_i^0$. Let us denote them by $\mathbb{P}_1, \ldots, \mathbb{P}_6$ so that $\mathbb{P}_1 \sqsubset \ldots \sqsubset \mathbb{P}_6$. The following holds:

$$\mathbb{F}_{i-1}^1 \sqsubset \mathbb{P}_1 \sqsubseteq \mathbb{F}_i^0 \sqsubset \mathbb{P}_2 \sqsubseteq \mathbb{A}_i \sqsubset \mathbb{B}_i \sqsubset \mathbb{P}_3 \sqsubseteq \mathbb{H}_i \sqsubset \mathbb{P}_4 \sqsubseteq \mathbb{C}_i \sqsubset \mathbb{D}_i \sqsubset \mathbb{P}_5 \sqsubseteq \mathbb{F}_i^1 \sqsubset \mathbb{P}_6.$$

$\{\mathbb{P}_1, \ldots, \mathbb{P}_6\}$ is π-closed. $\mathbb{P}_1, \mathbb{P}_3, \mathbb{P}_5 \in \mathcal{N}_{\mathbb{W}}$. $[\mathbb{P}_1, \mathbb{P}_2]_{\sqsubset}$,$[\mathbb{P}_3, \mathbb{P}_6]_{\sqsubset}$, and $[\mathbb{P}_4, \mathbb{P}_5]_{\sqsubset}$ cannot be π-closed, therefore there are only two possibilities: either $\pi(\mathbb{P}_1) = \mathbb{P}_4$, $\pi(\mathbb{P}_3) = \mathbb{P}_2$, and $\pi(\mathbb{P}_5) = \mathbb{P}_6$, or $\pi(\mathbb{P}_1) = \mathbb{P}_6$, $\pi(\mathbb{P}_3) = \mathbb{P}_4$, and $\pi(\mathbb{P}_5) = \mathbb{P}_2$.

Suppose that $\pi(\mathbb{P}_1) = \mathbb{P}_4$, $\pi(\mathbb{P}_3) = \mathbb{P}_2$, and $\pi(\mathbb{P}_5) = \mathbb{P}_6$. Since $[\mathbb{P}_1, \mathbb{P}_4]_{\sqsubset}$ is π-closed, in $(\mathbb{F}^1_{i-1}, \mathbb{D}_i]_{\sqsubset}$ the only elements for which we had not determined $\pi(\mathbb{A})$ are elements in $(\mathbb{F}^1_{i-1}, \mathbb{F}^0_i]_{\sqsubset}$ and in $(\mathbb{C}_i, \mathbb{D}_i]_{\sqsubset}$ with $t(\mathbb{A}) = p^j_{i-1}$ and with $t(\mathbb{A}) = q^j_{i-1}$. Notice that $t(\mathbb{D}_i) = p^m_{i-1}$ and $\mathbb{D}_i \in \mathcal{N}_{\mathbb{W}'}$.

If $i = 1$, then there are only two variants for $\pi(\mathbb{D}_i)$: one is $p_0^{m(l)}$ and the other one is $\mathbb{D}_1 p_0^{m(l)}$, where $l = 2$ or $l = 4$. Therefore, since the φ-join of $\mathbb{D}_1$ and $\mathbb{D}_1 p_0^{m(l)}$ is $\mathbb{F}^1_1 \in \mathcal{N}_{\mathbb{W}'}$, $\pi(\mathbb{D}_1) = p_0^{m(l)}$ and $[p_0^{m(l)}, \mathbb{D}_1]_{\sqsubset}$ is π-closed.

If $i > 1$, then there are four variants for $\pi(\mathbb{D}_i)$: $\mathbb{F}^1_{i-1} p^{m(l)}_{i-1}$, $\mathbb{D}_i p^{m(l)}_{i-1}$, where $l = 2$ or $l = 4$, $\mathbb{H}_{i-1} p^{m(2)}_{i-1}$, and $\mathbb{F}^0_{i-1} p^{m(-2)}_{i-1}$. The second variant is ruled out. If $\pi(\mathbb{D}_i) = \mathbb{H}_{i-1} p^{m(2)}_{i-1}$, then $\pi(\mathbb{D}_{i-1}) = \mathbb{D}_{i-1} p^{m(l)}_{i-2}$, where $l = 2$ or $l = 4$, and the φ-join of $\mathbb{D}_{i-1}$ and $\mathbb{D}_{i-1} p^{m(l)}_{i-2}$ is $\mathbb{F}^1_{i-1} \in \mathcal{N}_{\mathbb{W}'}$. If $\pi(\mathbb{D}_i) = \mathbb{F}^0_{i-1} p^{m(-2)}_{i-1}$ then since the segment $(\mathbb{F}^0_{i-1}, \mathbb{D}_i]_{\sqsubset}$ is φ-closed and π-closed, $\mathbb{G} \not\leq_\psi \mathbb{F}^0_{i-1} p^{m(-2)}_{i-1}$ for all $\mathbb{G} \notin (\mathbb{F}^0_{i-1}, \mathbb{D}_i]_{\sqsubset}$. But $\psi^2(\mathbb{D}_i) = \varphi(\pi(\mathbb{D}_i)) = \varphi(\mathbb{F}^0_{i-1} p^{m(-2)}_{i-1}) = \mathbb{F}^0_{i-1} \notin (\mathbb{F}^0_{i-1}, \mathbb{D}_i]_{\sqsubset}$. Therefore $\mathbb{D}_i \not\leq_\psi \varphi(\mathbb{D}_i) = \mathbb{H}_i p_i^{m(2)}$ and proof condition (4) is not satisfied. Therefore $\pi(\mathbb{D}_i) = \mathbb{F}^1_{i-1} p^{m(l)}_{i-1}$ and $(\mathbb{F}^1_{i-1}, \mathbb{D}_i]_{\sqsubset}$ is π-closed.

Therefore since $(\mathbb{F}^1_{i-1}, \mathbb{D}_i]_{\sqsubset}$ is π-closed and φ-closed, by Lemma 3 for $\mathbb{W}'_1$ there is π' satisfying proof conditions (1)-(4) and

$$\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \dots \Pi_{i-1} F_i(1) \dots F_n(t_n) \to G.$$

Suppose that $\pi(\mathbb{P}_1) = \mathbb{P}_6$, $\pi(\mathbb{P}_3) = \mathbb{P}_4$, and $\pi(\mathbb{P}_5) = \mathbb{P}_2$.

Here in $(\mathbb{F}^0_i, \mathbb{F}^1_i]_{\sqsubset}$ the only elements for which we had not determined $\pi(\mathbb{A})$ are elements in $(\mathbb{D}_i, \mathbb{F}^1_i]_{\sqsubset}$ and in $(\mathbb{F}^0_i, \mathbb{A}_i]_{\sqsubset}$ with $t(\mathbb{A}) = p^j_{i+1}$ and with $t(\mathbb{A}) = q^j_{i+1}$. Let $\mathbb{E} = \mathbb{F}^0_i p^{m(-2)}_{i+1} \in \mathcal{P}_{\mathbb{W}'}$.

There are only two variants for $\pi(\mathbb{E})$: one is $\mathbb{F}^0_i$ and the other one is $\mathbb{F}^1_i$. The φ-join of $\mathbb{E}$ and $\mathbb{F}^0_i$ is $\mathbb{F}^0_i \in \mathcal{N}_{\mathbb{W}}$. Therefore $\pi(\mathbb{E}) = \mathbb{F}^1_i$ and $(\mathbb{F}^0_i, \mathbb{F}^1_i]_{\sqsubset}$ is π-closed.

Therefore since $(\mathbb{F}^0_i, \mathbb{F}^1_i]_{\sqsubset}$ is π-closed and φ-closed, by Lemma 3 for $\mathbb{W}'_0$ there is π' satisfying proof conditions (1)-(4) and

$$\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \dots \Pi_{i-1} F_i(0) \dots F_n(t_n) \to G.$$

Lemma 14. *If* $\mathrm{L}^*(\backslash, /) \vdash \Pi_1 \dots \Pi_n \to G$, *then the formula* $c_1 \wedge \dots \wedge c_m$ *is satisfiable.*

Proof. Applying n times Lemma 13, we get that there exists $\langle t_1, \dots, t_n \rangle \in \{0,1\}^n$ such that $\mathrm{L}^*(\backslash, /) \vdash F_1(t_1) \dots F_n(t_n) \to G$. By Lemma 8 this means that $\langle t_1, \dots, t_n \rangle$ is a satisfying assignment for $c_1 \wedge \dots \wedge c_m$.

Since for all sequents $\mathrm{L}(\backslash, /) \vdash \Pi \to A \Rightarrow \mathrm{L}^*(\backslash, /) \vdash \Pi \to A$, Lemma 12 and Lemma 14 together give us Theorem 1.

Acknowledgements

I am most grateful to Prof. M. Pentus for his constant attention to my work.

References

1. Aarts, E., Trautwein, K.: Non-associative Lambek categorial grammar in polynomial time. Mathematical logic Quarterly 41, 476–484 (1995)
2. Buszkowski, W.: The equivalence of unidirectional Lambek categorial grammars and context-free grammars. Zeitschrift für mathematische Logik und Grundlagen der Mathematik 31(4), 369–384 (1985)
3. de Groote, P.: The non-associative Lambek calculus with product in polynomial time. In: Murray, N.V. (ed.) TABLEAUX 1999. LNCS, vol. 1617, pp. 128–139. Springer, Heidelberg (1999)
4. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65(3), 154–170 (1958)
5. Pentus, M.: Lambek calculus is NP-complete. Theoretical Computer Science 357(1–3), 186–201 (2006)
6. Pentus, M.: Lambek grammars are context free. In: Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science, pp. 429–433 (1993)
7. Savateev, Y.: Lambek grammars with one division are decidable in polynomial time. In: Hirsch, E.A., et al. (eds.) Computer Science – Theory and Applications. LNCS, vol. 5010, pp. 273–282. Springer, Heidelberg (2008)

Games on Multi-stack Pushdown Systems

Anil Seth

Department of Computer Science & Engg.
I.I.T. Kanpur, Kanpur 208016, India
seth@cse.iitk.ac.in

Abstract. Bounded phase multi-stack pushdown automata have been studied recently. In this paper we show that parity games over bounded phase multi-stack pushdown systems are effectively solvable and winning strategy in these games can be effectively synthesized. We show some applications of our result, including a new proof of a known result that emptiness problem for bounded phase multi-stack automata is decidable.

1 Introduction

A multi-stack pushdown system (*mpds*) has a finite set of control states and a fixed number of stacks. The transition function of a *mpds* takes as input its control state and topmost symbols of each stack and may (nondeterministically) do a push or a pop operation on any stack along with a possible change in control state of *mpds*. A *mpds* obviously generalizes a pushdown system *pds* as it can have more than one stack. While pushdown systems can be used to model sequential recursive programs, multi-stack pushdown systems can be used to model a class of programs with both recursion and threads. Each thread has its own stack for its procedures calls and communication among threads is through the common finite states of *mpds*. Model checking of programs with threads is an important problem and there have been several recent works, see [1,3,4,5,6], in the area of model checking *mpds* and their variants. Some restrictions however are needed to be imposed on *mpds* to get effectively checkable properties of the model, as even simple properties such as reachability from one configuration to another are undecidable for unrestricted *mpds*. A restriction considered in [4,5] is bounded context switching. In a k context switching *mpds* we consider only those runs of *mpds* which can be divided into k stages, where each stage is a consecutive sequence of moves from the run in which push and pop operations are performed only in one stack. While this seems a strong restriction, it has been useful in practice and reachability analysis with this restriction has helped uncover some errors. Bounded context switching *mpds* admit effective global reachability analysis also. For a *mpds* M and a regular set $\mathcal{C}$ of configurations of M, $pre^*(\mathcal{C})$, the set of configurations of M from which a configuration in $\mathcal{C}$ can be reached by M, is shown to be regular in [5]. Similarly $post^*(\mathcal{C})$, the set of configurations to which a configuration in $\mathcal{C}$ can be reached by M, is also shown to be regular in [5].

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 395–408, 2009.

In [1], a more liberal version of *mpds* called bounded phase systems have been considered. In a $k-$phase bounded *mpds* only those runs of *mpds* are considered which can be divided into k stages where each stage is a consecutive sequence of moves from the run in which *pop* operations are performed only in one stack (push operations can be performed on any stack in a single phase). The class of bounded phase *mpds* strictly includes the class of bounded context switching *mpds*. In [1] emptiness problem of bounded phase multi-stack automata (*mpda*) is shown to be decidable. Bounded phase *mpds* have also been used in model checking of concurrent queues [3] and bounded phase multi-stack pushdown transducers have been used to give an infinite automaton characterization of complexity class of problems solvable in double exponential time, 2ETIME [2].

In model checking, it is often needed to consider richer properties than reachability. This can be done meaningfully over restricted mpds also. For example, existence of a bad infinite path in a bounded phase *mpds* implies the existence of the same for the unrestricted *mpds* also. A general way to specify verification problems is through infinite games [9], where evaluating a formula in a model is equivalent to deciding if a player has a winning strategy in a game. For example, evaluating a modal mu-calculus formula on a transition system can be reduced to solving a parity game on a graph closely related to the transition system. Games can also be used to model reactive systems naturally. Two player parity games over single stack pushdown systems (pds) have been studied in [9] where it is shown that these games can be effectively solved. That is, there is an algorithm which from the description of a game can determine which player has a winning strategy in the game starting from the initial configuration of *pds*.

Two player games over *mpds* have not been studied so far. In this paper, we study two player reachability and parity games over bounded phase *mpds* and show them to be effectively solvable. Our solution is based on a fundamental technique of Walukiewicz [9] which shows how to reduce a parity game on a pushdown system to a parity game over finite state space. In [9] each time a symbol is pushed in the stack, a set of states (along with priorities) is guessed by player 0, the game now divides into two parts. In the first subgame player 1 verifies that if the symbol is popped then it is in one of the guessed states, in the second game it is verified that if the pushed symbol is popped satisfying the guessed conditions then the game is winning for player 0. This does not let the stack grow in any subgame, only topmost symbol of the stack needs to be kept in a game state along with some bounded auxiliary information, thus resulting in a finite state game.

We extend this technique to the case of bounded phase *mpds*. To begin with, we need to store the topmost symbol of each stack. The main difficulty in case of more than one stack is that when a symbol in some stack i is pushed it is not sufficient to guess the states and minimum priorities visited for popping this symbol. The contents of stack j, $j \neq i$, can change in an arbitrary way between a push and the corresponding pop in stack i. For example, we may have a sequence like $push_i$, $push_j$ $push_j$ pop_i, where subscript indicates the

stack on which the operation is performed. In general, we may have an arbitrary number of push operations to stack j, $j \neq i$ between a matching push and pop operation in stack i. The information about contents of stack j at the time of pop in stack i is needed to simulate the *mpds* transitions after the pop_i. The important issue is to see how much information about stack j is needed, will a bounded amount of information suffice? Let us examine this, we need to know at least the topmost symbol of stack j, say γ', at the time of pop_i. In addition to this, to abstract out the contents of stack j below γ', we may need something like popping condition for γ' after pop_i. But this seems circular as in identifying the popping condition for $push_i$ we need to know the popping condition for γ' in stack j, $j \neq i$. Further, the popping condition for γ' of stack j, $j \neq i$, needs to be guessed at the time of $push_i$ only. The apparent circularity may be removed by looking at how the phase changes after various *pop* operations involved. We formulate these conditions recursively using recursion on the number of phases. This is the main technical contribution in our reduction of bounded phase *mpds* game to a finite state game. The size of the FSG, we get is rather huge. It is a tower of exponentials of height k, the number of phases of *mpds*. This is also the complexity of our decision procedure to solve *mpds* game.

In [9] it is also shown that the winning strategy in pds game can be executed by a pushdown automaton. This extends naturally to our setting, we show that the winning strategy in bounded phase *mpds* game can be executed by a bounded phase *mpda.* As a special case of our general game, we consider one player reachability game, in which all configurations belong to player 0 only and the winning condition is reaching some specified control state. Existence of winning strategy for player 0 in this game is equivalent to the specified control state being reachable from the initial configuration of *mpds*. In this case the size of our finite state game reduces to double exponential in the number of phases allowed. This special case gives us an alternative proof of one of the main results in [1] that the emptiness problem of bounded phase *mpda* is decidable. Our proof uses quite different technique than the ones used in [1]. In [1] a translation of bounded phase words into finite trees is defined such that the image of this translation is definable in monadic second order logic (MSO) over finite trees. Decidability of MSO on the class of finite trees is then used in [1] to derive the result. The complexity of our decision procedure matches the optimal complexity bound obtained in [3]. Further, our strategy synthesis result when specialized to one player reachability game, can be used to generate counter-examples in automatic verification of safety properties of bounded phase *mpds*.

2 Preliminaries

Definition 1. *A multi-stack pushdown system (mpds) is given as a tuple* $(Q, \Gamma, l, \delta, q_0)$*, where* Q *is a finite set of states,* l *is the number of stacks,* Γ *is the stack alphabet and* q_0 *is the initial state. The transition function* δ *is given as* $\delta = \delta_i \cup \delta_r \cup \delta_e$*, where*

- $\delta_e \subseteq Q \times \Gamma^l \times Q \times [1 \dots l] \times \Gamma$,
- $\delta_i \subseteq Q \times \Gamma^l \times Q \times [1 \dots l] \times \Gamma$,
- $\delta_r \subseteq Q \times \Gamma^l \times [1 \dots l] \times Q$.

($\delta_e, \delta_i, \delta_r$ represent exchange, push and pop operations respectively). An mpds *operation depends on its control state and topmost symbols of all its stacks. Γ has a special symbol $\perp$ for marking bottom of a stack. $\perp$ can't be pushed, popped or exchanged by any other symbol. For instance, if $(q, \bar{\gamma}, q', j, \gamma) \in \delta_e$, where $\bar{\gamma} = \gamma_1 \dots \gamma_l$ and $\gamma_j = \perp$ then $\gamma = \perp$. Also, if $(q, \bar{\gamma}, i, q') \in \delta_r$ then $\gamma_i \neq \perp$.*

Definition 2. *A configuration of multi-stack pushdown system $(Q, \Gamma, l, \delta, q_0)$ is a tuple $(q, s_1, \dots, s_l)$, where $q \in Q$ and $s_i \in \{\perp\} \times (\Gamma - \{\perp\})^*$, for $1 \leq i \leq l$. One step transition $\xrightarrow{t}$ on configurations of mpds is defined as below, where $\bar{\gamma} = \gamma_1 \dots \gamma_l$.*

- $(q, s_1.\gamma_1, \dots, s_l.\gamma_l) \xrightarrow{t} (q', s'_1, \dots, s'_l)$ *if* $t = (q, \overline{\gamma}, q', i, \gamma) \in \delta_e$, $s'_i = s_i.\gamma$ *and* $s'_j = s_j.\gamma_j$ *for* $j \neq i, 1 \leq j \leq l$.
- $(q, s_1.\gamma_1, \dots, s_l.\gamma_l) \xrightarrow{t} (q', s'_1, \dots, s'_l)$ *if* $t = (q, \overline{\gamma}, q', i, \gamma) \in \delta_i$, $s'_i = s_i.\gamma_i.\gamma$ *and* $s'_j = s_j.\gamma_j$ *for* $j \neq i, 1 \leq j \leq l$.
- $(q, s_1.\gamma_1, \dots, s_l.\gamma_l) \xrightarrow{t} (q', s'_1, \dots, s'_l)$ *if* $t = (q, \overline{\gamma}, i, q') \in \delta_r$, $s'_i = s_i$ *and* $s'_j = s_j.\gamma_j$ *for* $j \neq i,\ 1 \leq j \leq l$.

The initial configuration of mpds *is defined as* $(q_0, \perp, \dots, \perp)$.

Definition 3. *A multi-step transition between configurations of mpds, on say sequence $t_1 t_2 \dots t_n$ of* mpds *moves, $c \overset{t_1 t_2 \dots t_n}{\twoheadrightarrow} d$ is defined as follows. $c \overset{t_1 t_2 \dots t_n}{\twoheadrightarrow} d$ iff either $n = 0$ and $c = d$ or there is a c' s.t. $c \xrightarrow{t_1} c'$ and $c' \overset{t_2 \dots t_n}{\twoheadrightarrow} d$. We write $c \twoheadrightarrow d$ for a multi-step transition from c to d when the sequence of* mpds *moves is not relevant.*

Definition 4. *Let c_0 be the initial configuration of a* mpds *and let c be a configuration reached from c_0 by a sequence $t_1 t_2 \dots t_n$ of* mpds *moves. If the topmost symbol of stack i is $\gamma \neq \perp$, then the push operation corresponding to the topmost symbol of stack i is defined as the last unmatched push operation in stack i in the sequence $t_1 t_2 \dots t_n$.*

The previous definition is usually understood without explicitly mentioning it. It may be noted however that because of exchange operations the push operation corresponding to γ may actually have pushed a symbol γ', $\gamma' \neq \gamma$ on the stack. For example, after sequence $push_1^{\gamma_1}$, $exch_1^{\gamma_1,\gamma}$, (where $push_i^{\gamma}$ means pushing symbol γ to stack i, $exch_i^{\gamma,\gamma'}$ means replacing the topmost symbol γ of the stack i by γ') the topmost symbol on stack 1 is γ, but it corresponds to $push_1^{\gamma_1}$ operation. Similarly, in sequence $push_1^{\gamma_1}$, $exch_1^{\gamma_1,\gamma}$, pop_1, the last pop matches the first push operation $push_1^{\gamma_1}$.

Definition 5. *A configuration d of multi-stack pushdown system (Q, Γ, l, δ) is reachable from configuration c in m-phases if there are $\alpha_1, \ldots, \alpha_m$ where each α_i is a sequence of* mpds *moves with pop moves from at most one stack and $c \overset{\alpha_1}{\twoheadrightarrow} c_1 \overset{\alpha_2}{\twoheadrightarrow} c_2 \ldots \overset{\alpha_m}{\twoheadrightarrow} c_m = d$.*

We assume the reader to be familiar with standard notions of two player parity games, such as game graph, plays, a winning strategy and parity winning condition, see [12].

A 2-player k-phase *mpds* parity game is given as $(\mathcal{H}, Q_0, Q_1, M, \Omega, k)$, where $\mathcal{H} = (Q, \Gamma, l, \delta, q_0)$ is an *mpds*, $Q = Q_0 \oplus Q_1$ is a partition of states in player 0 and player 1, M is a finite set of priorities and $\Omega : Q \to M$ is a priority assignment to each state in Q.

Vertices of our game graph are configurations of *mpds*. A vertex $(q, -, \ldots, -)$ belongs to player i iff $q \in Q_i$. priority of a vertex $(q, -, \ldots, -)$ is defined as $\Omega(q)$. A player can move from c to c' only if $c \to c'$. A play is a sequence of legal moves starting from the initial configuration. A phase in a play is a consecutive sequence of moves such that in this sequence elements from at most one stack are popped (though in a single phase elements may be pushed to any stack). All plays in this game are $k - phase$ bounded. That is a player can not make a move that takes the play into $(k+1)^{th}$ phase.

Remark1: Some authors take a play to be a sequence of moves which can not be extended further, we use the descriptor maximal play for this and take a play to be any sequence of moves as defined above.

Remark2: We could make the game graph more standard by taking its vertices as triples (c, p, r) where c is an *mpds* configuration $p \leq k$ is the phase in play so far and $r \leq l$ is the number of stack popped last. There is an edge from (c, p, r) to (c', p', r') iff $c \to c'$ and as a result of this transition the phase changes from p to p' and the number of the last popped stack changes from r to r' in the *mpds* configuration.

Winning condition for a maximal play (play which can not be extended further) ρ is defined as follows. If ρ is finite then the player whose turn it is to move at the last vertex of ρ loses. If ρ is infinite then a priority $i \in M$ is said to be visited infinitely often iff there are infinitely many vertices with priority i in ρ. ρ is winning for player 0 iff the minimum, among the set of priorities visited infinitely often in ρ, is even.

Informally, having a winning strategy for player i, means that regardless of player $(1-i)$'s moves, player i can always play a move such that he wins the resulting play. We will always consider games which start in a predefined initial configuration. A game is called winning for player i if player i has a winning strategy in it starting from the initial configuration.

Given a winning strategy τ for player 0, in game $\mathcal{G}$, by a τ-play we mean a play of $\mathcal{G}$ in which all moves of player 0 are according to τ. For configurations c, c' of $\mathcal{G}$, $c \overset{\tau}{\to} c'$ and $c \overset{\tau}{\twoheadrightarrow} c'$ mean that c' is reachable from c in a τ-play in one move or in an arbitrary number of moves respectively.

3 Reducing *MPDS* Game to Finite State Game

3.1 Intuitive Idea

Let $\mathcal{H} = (Q, \Gamma, l, \delta, q_0)$ be a *mpds* and let $\mathcal{G} = (\mathcal{H}, Q_0, Q_1, max, \Omega : Q \to M)$ be a game structure on $\mathcal{H}$, where $Q = Q_0 \oplus Q_1$ and $M = \{0, \ldots, max\}$ is the set of priorities assigned to vertices of the game. We use below notation $\overline{T}$ for a sequence $T_1, \ldots, T_l$ and $\overline{T}[C/i]$ for sequence $T_1, \ldots, T_{i-1}, C, T_{i+1}, \ldots, T_l$, which is the same as $\overline{T}$ except at i^{th} position where it is C. We follow [8] in presentation of our finite state game.

Most important vertices of the finite state game (FSG) are of the form $Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m})$, where $q \in Q$, $p \in [1, k]$, $r \in [0, l]$, $\overline{\gamma} = \gamma_1 \ldots \gamma_l$ with each $\gamma_i \in \Gamma$, $\overline{B} = B_1, \ldots, B_l$ with each $B_i \in \tau_i$ and $\overline{m} = m_1 \ldots m_l$ with $m_i \in \{0 \ldots max\}$. Intuitively the vertex $Check(q, p, s, \overline{\gamma}, \overline{B}, \overline{m})$ asserts that

- q is the state of the configuration.
- p is the current phase.
- r is the number of stack from which last pop operation was done (initially it is set to 0).
- γ_i is the topmost symbol of stack i.
- m_i is the minimum priority seen since the push operation corresponding to the topmost symbol of stack i (if topmost symbol is $\perp$, which is never pushed, then $m_i = 0$).
- $B_i \subseteq \cup_{j=1}^{k} N_{i,j}$ gives constraints to be met on popping the topmost element of stack i (if stack i is empty then $B_i = \emptyset$).

When an element of stack i is popped, the resulting configuration, say d, is in a phase belonging to $[1, k]$. $N_{i,j}$ is the set of conditions related to pop operations of stack i which result in configurations of phase j on popping. In a FSG play for each element pushed in stack i a subset of all popping scenarios $\cup_{j=1}^{k} N_{i,j}$ is guessed. $N_{i,j}$ is defined below simultaneously for $1 \leq i \leq l$ using recursion on j, starting from $j = k$ going down to $j = 1$.

Definition 6. *In this definition we assume that* $q, \overline{m}, \overline{\gamma}$ *range over* Q, M^l, Γ^l *respectively and* p *ranges over* $[1, l]$.

$N_{i,k} = \{(a_1 ... a_{i-1}, (q, k, \overline{\gamma}, \overline{m}), a_{i+1} ... a_l) \mid a_p = \emptyset \ p \neq i\}$
For j, $k > j \geq 1$,
$N_{i,j} = \{(a_1 ... a_{i-1}, (q, j, \overline{\gamma}, \overline{m}), \ a_{i+1} ... a_l) \mid a_p \subseteq \cup_{r=j+1}^{k} N_{p,r} \ p \neq i\}$.

Each element $u \in N_{ij}$ describes a scenario for popping an element of stack i. Assume that the element is popped in *mpds* configuration c and the resulting *mpds* configuration (after pop) is d. The tuple $(q, j, \overline{\gamma}, \overline{m})$ in a scenario u stipulates that $\overline{\gamma} = \gamma_1 \ldots \gamma_l$ and $\overline{m} = m_1 \ldots m_l$, where for $1 \leq r \leq l$, γ_r is the topmost symbol of stack r in c and m_r is the value in c whose interpretation is as described above. q is the *mpds* state in d and j is the phase of d. a_t for $1 \leq t \leq l$, $t \neq i$, is a set of popping conditions for the topmost symbol γ_t of stack t in configuration d. That is a_t stands for conditions in which γ_t can be

popped. This uses recursively the conditions defined for popping a symbol. Note that after popping stack i in d a pop in stack t will result in configuration of phase $> j$. This ensures well defined nature of recursion. Also note that we use a_t as a set of scenarios, instead of a single scenario for popping of γ_t in some configuration after d. This is necessary in a two player game, because in a two player game player 0 can only guarantee that the resulting *mpds* configuration be one from a set (rather than a uniquely specified) of *mpds* configurations.

In the definition of $N_{i,k}$, a_t for $t \neq i$, is taken to be $\emptyset$ because after popping the symbol in stack i phase k is reached and no other stack can be popped, so popping conditions for topmost symbol of stack t is not of any use now.

3.2 The Finite State Game (FSG)

Each *mpds* transition gives rise to some FSG transitions. We group transitions of FSG according to *mpds* transitions (shown in bold) which give rise to them.

1. $(\mathbf{q}, \overline{\gamma}, \mathbf{q'}, \mathbf{i}, \gamma') \in \delta_{\mathbf{e}}$, $1 \leq i \leq l$.
   (a) $Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m}) \rightarrow Check(q', p, r, \overline{\gamma}[\gamma'/i], \overline{B}, \overline{m'})$, where $m'_j = min(m_j, \Omega(q'))$, for $1 \leq j \leq l$.
2. $(\mathbf{q}, \overline{\gamma}, \mathbf{q'}, \mathbf{i}, \gamma') \in \delta_{\mathbf{i}}$, $1 \leq i < l$.
   (a) $Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m}) \rightarrow Push_i(p, r, \overline{\gamma}, \overline{B}, \overline{m}, q', \gamma')$
   (b) $Push_i(p, r, \overline{\gamma}, \overline{B}, \overline{m}, q', \gamma') \rightarrow Claim_i(p, r, \overline{\gamma}, \overline{B}, \overline{m}, q', \gamma', C)$, for all $C \subseteq \cup_{j=1}^{k} N_{i,j}$.
   (c) $Claim_i(p, r, \overline{\gamma}, \overline{B}, \overline{m}, q', \gamma', C) \rightarrow Check(q', p, r, \overline{\gamma}[\gamma'/i], \overline{B}[C/i], \overline{m'})$,

$$where \quad m'_j = \begin{cases} min(m_j, \Omega(q')) & if\ j \neq i \\ \Omega(q') & if\ j = i \end{cases}$$

   (d) To check the game after corresponding pop_i operation.
   $Claim_i(p, r, \overline{\gamma}, \overline{B}, \overline{m}, q', \gamma', C) \rightarrow Jump_i(q'', j, \overline{\gamma}, \overline{\gamma''}, \overline{m'}, \overline{B'}, \overline{m})$
   for all $(a_1 ... a_{i-1}, (q'', j, \overline{\gamma''}, \overline{m'}),\ a_{i+1} ... a_l) \in C$
   where $\overline{B'} = (a_1 ... a_{i-1}, B_i, a_{i+1} ... a_l)$.
   (e) $Jump_i(q'', j, \overline{\gamma}, \overline{\gamma''}, \overline{m'}, \overline{B'}, \overline{m}) \rightarrow Check(q'', j, i, \overline{\gamma''}[\gamma_i/i], \overline{B'}, \overline{m''})$,

$$where \quad m''_j = \begin{cases} min(m'_j, \Omega(q'')) & if\ j \neq i \\ min(m_i, m'_i, \Omega(q'')) & if\ j = i \end{cases}$$

3. $(\mathbf{q}, \overline{\gamma}, \mathbf{i}, \mathbf{q'}) \in \delta_{\mathbf{r}}$, $1 \leq i \leq l$.
   (a) $Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m}) \rightarrow Win_0$ if $\overline{C}[(q', p', \overline{\gamma}, \overline{m})/i] \in B_i$

(b) $Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m}) \rightarrow Win_1$ if $\overline{C}[(q', p', \overline{\gamma}, \overline{m})/i] \notin B_i$

$$where \quad p' = \begin{cases} 1 & if\ r = 0 \\ p & if\ r = i \\ p+1 & if\ r \neq i \end{cases},$$

$$p' \leq k \text{ and } \overline{C} = \begin{cases} \overline{B} & if\ \ p' < k \\ \overline{\emptyset} & if\ \ p' = k \end{cases}.$$

Priority of vertex v in FSG, denoted by $\lambda(v)$, is defined as follows. $\lambda(Check(q, \ldots)) = \Omega(q)$, $\lambda(Push_i(\ldots)) = \lambda(Claim_i(\ldots)) = max$ and $\lambda(Jump_i(q, j, \overline{\gamma}, \overline{\gamma'}, \overline{n}, \overline{B}, \overline{m})) = n_i$, where $\overline{n} = n_1 \ldots n_l$. Vertex $Check(q, \ldots)$ belongs to player j, $j \in \{0, 1\}$, iff $q \in Q_j$. Vertices $Push_i(...)$, belong to player 0 whereas vertices $Claim_i(...)$, belong to player 1, for $0 \leq i < n$. Each vertex $Jump_i(...)$ has a single outgoing edge so it is immaterial which player is assigned to these vertices.

4 Relating Winning in *MPDS* Game and the FSG

Our main theorem is the following.

Theorem 1. *A* mpds *game is winning for player* 0 (*from initial configuration* $(q_0, \bot_{\mathcal{S}}, \ldots, \bot_{\mathcal{S}})$) *iff FSG is winning for player* 0 (*from initial configuration* $Check(q_0, 1, 0, \overline{\bot}, \overline{\emptyset}, \overline{0})$). *Further, if* mpds *game is winning for player* 0 *then player* 0 *has a winning strategy in* mpds *game that is computable by a multi-stack automaton.*

Proof. We present the construction of strategy automaton below, this is used in proving the direction that a winning strategy for player 0 in FSG implies a winning strategy in *mpds* game. The detailed correctness of this construction as well as the other direction of the theorem that a winning strategy for player 0 in mpds game implies a winning strategy in FSG, are given in full version of this paper. Idea of the proofs is similar to that in [8], but we need to deal with more involved cases as we argue for multi-stack pushdown systems. □

4.1 Strategy Automaton

Assuming that there is a winning strategy for player 0 in FSG from $Check(q_0, 1, 0, \overline{\bot}, \overline{\emptyset}, \overline{0})$, we design a l stack pushdown automaton $\mathcal{S}$ which executes a winning strategy τ of player 0 in *mpds* game from *mpds* configuration $(q_0, \bot_{\mathcal{S}}, \ldots, \bot_{\mathcal{S}})$.

Fix a history free winning strategy σ for player 0 in FSG from configuration $Check(q_0, 1, 0, \overline{\bot}, \overline{\emptyset}, \overline{0})$ (such a strategy exists in any parity game, see [12]). Using σ, we design a deterministic l stack pushdown automaton $\mathcal{S}$ as follows. Apart from the stacks, $\mathcal{S}$ has an input and an output tape. $\mathcal{S}$ reads moves of player 1

from the input tape and outputs moves of player 0 on the output tape. Structure of $\mathcal{S}$'s stacks, at any point in play, is the same as that of *mpds* stacks at that point. For a symbol $\gamma \in \Gamma$ in stack i of mpds, $\mathcal{S}$ stores at the corresponding position in its stack i a tuple of the form (γ, B, m), where $B \subseteq \cup_{j=1}^{k} N_{i,j}$ and $m \in \{0, 1, \ldots, max\}$. The additional information (B, m) in stacks of $\mathcal{S}$ records relevant information about the FSG play being simulated. Bottom of stack marker for $\mathcal{S}$ is defined to be $\perp_{\mathcal{S}} = (\perp, \emptyset, 0)$. Control state of $\mathcal{S}$ is of the form (q, p, r), where q is the current state in *mpds* game, p is the phase and r is the stack from which *mpds* play can pop in phase p.

Configuration of $\mathcal{S}$ is defined as its state along with the contents of its stacks. It subsumes the current *mpds* configuration in it. We define a function f from configurations of $\mathcal{S}$ to vertices of FSG. If c is a configuration of $\mathcal{S}$ in which state of $\mathcal{S}$ is (q, p, r) and top of the stack j symbol is (γ_j, B_j, m_j), for $1 \leq j \leq l$, then $f(c) = Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m})$. In strategy execution the following invariant is maintained: if $\mathcal{S}$ is in configuration c then there is a σ play from $f(c_0)$ to $f(c)$, where c_0 is the initial configuration of $\mathcal{S}$.

Suppose $\mathcal{S}$ is in configuration c as above. The next move in the *mpds* play is either read from the input tape (if $q \in Q_1$) or $\mathcal{S}$ mimics the σ move (if $q \in Q_0$) from the corresponding FSG configuration $f(c) = Check(q, p, r, \overline{\gamma}, \overline{B}, \overline{m})$. (Moves from an *mpds* configuration and from the corresponding $Check(\ldots)$ vertex in FSG are in $1-1$ correspondence by design of FSG.)

4.2 Operation of $\mathcal{S}$

Initially, $\mathcal{S}$ is in state $(q_0, 1, 0)$ and each stack of $\mathcal{S}$ is $\perp_{\mathcal{S}}$. At any arbitrary point in *mpds* play if $\mathcal{S}$ is in configuration c in which state of $\mathcal{S}$ is (q, p, r) and top of the stack j symbol is (γ_j, B_j, m_j), for $1 \leq j \leq l$. Then for each *mpds* move played in the *mpds* game, we describe actions performed by $\mathcal{S}$. Stack and state updation of $\mathcal{S}$ below is grouped by the moves of *mpds* regardless of whichever player plays.

[I] $(q, \overline{\gamma}, q', i, \gamma') \in \delta_e$, $1 \leq i \leq l$.

$\mathcal{S}$ sets top of the stack i symbol to $(\gamma', B_i, min(m_i, \Omega(q')))$ and top of the stack j (for $j \neq i$) symbol to $(\gamma_j, B_j, min(m_j, \Omega(q')))$ and changes its control state to (q', p, r).

[II] $(q, \overline{\gamma}, q', h, \gamma') \in \delta_i$, $1 \leq h \leq l$.

Let $C \subseteq \cup_{t=1}^{k} N_{h,t}$ be as in transition (2.b) in corresponding σ play. $\mathcal{S}$ pushes $(\gamma', C, \Omega(q'))$ on stack h, topmost symbol of the stack j (for $j \neq h$) is changed to $(\gamma_j, B_j, min(m_j, \Omega(q')))$ and $\mathcal{S}$ changes its control state to (q', p, r).

[III] $(q, \overline{\gamma}, i, q') \in \delta_r$, $1 \leq i \leq l$.

$\mathcal{S}$ pops the topmost symbol from stack i. Let the resulting topmost symbol in stack i (after the above pop) be $(\gamma', D,\ n)$. $\mathcal{S}$ modifies it to $(\gamma', D, min(m_i, n, \Omega(q')))$. $\mathcal{S}$ modifies the topmost symbol of

stack j (for $j \neq i$) to $(\gamma_j, B_j, min(m_j, \Omega(q')))$ and enters control state (q', p', i), where

$$p' = \begin{cases} 1 & if\ r = 0 \\ p & if\ r = i \\ p+1 & if\ r \neq i \end{cases}.$$

4.3 Complexity of Solving the Game

By the reduction in section 3, to solve the *mpds* game it suffices to solve the associated FSG. In this section we estimate the size of FSG and the complexity of solving it. Let us begin by defining a class of functions $exp_n(m)$ iteratively as follows.

$exp_1(m) = 2^m$ and for $n \geq 1$, $exp_{n+1}(m) = 2^{exp_n(m)}$.

Roughly, $exp_n(m)$ is a tower of exponentials of height n. Let $\mathcal{H}$ be an *mpds* and $\mathcal{G}$ be an *mpds* game on $\mathcal{H}$ as in section 3.1. For a set A, we let $|A|$ denote its cardinality. Then by definition 6,

$|N_{i,k}| = |Q|.k.|M|^l.|\Gamma|^l$, for all i.

$|N_{i,j}| \leq t.(\prod_{p=1}^{l} 2^{\Sigma_{r=j+1}^{k}|N_{p,r}|})$, for $1 \leq j < k$ where $t = |Q|.k.|M|^l.|\Gamma|^l$.

$|N_{i,j}| \leq t.(\prod_{p=1}^{l} 2^{k|N_{p,j+1}|})$, using $|N_{p,j+1}| > |N_{p,r}|$ for $r > j+1$

$|N_{i,j}| \leq t.(2^{l.k|N_{i,j+1}|})$, because by symmetry $|N_{p,j+1}| = |N_{i,j+1}|$, for $1 \leq p \leq l$.

$|N_{i,j}| \leq (2^{c.l.k|N_{i,j+1}|})$ for a constant c, using $m.2^m \leq 2^{2m}$.

This leads to $|N_{i,1}| = exp_{k-1}(O(z))$ and $|B_1| = exp_k(O(z))$, where $z = l.k^2.|Q|.|M|^l.|\Gamma|^l$. The number of vertices in FSG is therefore $exp_k(O(z))$.

It is known that a game graph with n vertices, m edges and d priorities can be solved in time $O(m.n^d)$, see [10].

Number of edges in FSG is bounded by $[exp_k(O(z))]^2$, which is the same as $exp_k(O(z))$. Number of distinct priorities in FSG is $|M|$. It follows that our FSG can be solved and winning strategy can be constructed in time bounded by $exp_k(O(z.|M|))$, where $z = l.k^2.|Q|.|M|^l.|\Gamma|^l$ as mentioned above.

5 One Player Case

Let $\mathcal{H}$ be an *mpds* and $\mathcal{G}$ be an *mpds* game on $\mathcal{H}$ as in section 3.1. In this section, we consider the special case when all configurations belong to player 0, that is $Q = Q_0$. In this case popping conditions in definition 6 can be simplified as follows.

Definition 7. *In this definition we assume that* $q, \overline{m}, \overline{\gamma}$ *range over* Q, M^l, Γ^l *respectively and* p *ranges over* $[1, l]$.

$N_{i,k} = \{(a_1...a_{i-1}, (q, k, \overline{\gamma}, \overline{m}), a_{i+1}...a_l) \mid a_p = \emptyset\ p \neq i\}$

For j, $k > j \geq 1$,

$N_{i,j} = \{(a_1...a_{i-1}, (q, j, \overline{\gamma}, \overline{m}),\ a_{i+1}...a_l) \mid a_p \in \cup_{r=j+1}^{k} N_{p,r}\ \ p \neq i\}$.

$B_i \in \cup_{j=1}^{k} N_{i,j}$

Complexity analysis in this case becomes:

$|N_{i,k}| = |Q|.k.|M|^l.|\Gamma|^l$, for all i.

$|N_{i,j}| \leq \prod_{p=1}^{l}(\Sigma_{r=j+1}^{k}|N_{p,r}|)$, for $1 \leq j < k$.

$|N_{i,j}| \leq (\prod_{p=1}^{l} k.|N_{p,j+1}|)$ using $|N_{p,j+1}| > |N_{p,r}|$ for $r > j+1$.

$|N_{i,j}| \leq (k.|N_{i,j+1}|)^l$, because by symmetry $|N_{p,j+1}| = |N_{i,j+1}|$, for $1 \leq p \leq l$.

This leads to $|N_{i,1}| = z^{l^{O(k)}}$ and $|B_1| = z^{l^{O(k)}}$, for $l \geq 2$ where $z = |Q|.|M|.|\Gamma|$.

The number of vertices in FSG is therefore $z^{l^{O(k)}}$ and it can be solved in time $z^{|M|.(l^{O(k)})}$, where $z = |Q|.|M|.|\Gamma|$.

5.1 Bounded Phase Multi-stack Pushdown ω-Automata

We may also consider bounded phase multi-stack pushdown ω-automata on infinite words.

Definition 8. *A bounded phase multi-stack pushdown parity ω-automaton is given as a tuple $(Q, \Sigma, \Gamma, l, k, \delta, q_0, M, \Omega)$, where Q is a finite set of states, Σ is an input alphabet and k is a bound on the number of phases. Γ, q_0, l are the same as in definition 1. $\delta = \delta_i \cup \delta_r \cup \delta_e$ is also the same as in definition 1,2 except that each transition of the automaton also depends on the current symbol being read from the input tape apart from the state and the topmost symbols of all stacks. Therefore*

- $\delta_e \subseteq Q \times \Sigma \times \Gamma^l \times Q \times [1 \dots l] \times \Gamma$,
- $\delta_i \subseteq Q \times \Sigma \times \Gamma^l \times Q \times [1 \dots l] \times \Gamma$,
- $\delta_r \subseteq Q \times \Sigma \times \Gamma^l \times [1 \dots l] \times Q$.

A configuration of this automaton is the same as a *mpds* configuration along with position of input head on the input tape. With each move the input head moves one position to the right. A run of this automaton is a sequence of configurations starting with the initial configuration and in which for any two consecutive configurations the successor configuration is a result of some $\delta-$transition on the predecessor configuration. A run is accepting if the number of phases in it are $\leq k$ and it satisfies the parity acceptance condition given by Ω. We have the following theorem about such automata.

Theorem 2. *Emptiness problem for bounded phase multi-stack pushdown $\omega-$automata with parity acceptance condition is decidable in time $(|Q|.|M|.|\Gamma|)^{|M|.(l^{O(k)})}$, where Q, Γ, l, k, M are as in definition 8 and $l \geq 2$.*

Proof. Consider a bounded phase multi-stack ω-automaton $\mathcal{M}$ as in definition 8. We erase all input symbols from transitions of $\mathcal{M}$ and let all states belong to player 0. This gives a one player *mpds* game. Winning in this game for player 0 is equivalent to $\mathcal{M}$ having an accepting run on some input, that is $\mathcal{L}(\mathcal{M}) \neq \emptyset$. By section 5, this *mpds* game can be solved in time $(|Q|.|M|.|\Gamma|)^{|M|.(l^{O(k)})}$. $\square$

5.2 Reachability in Bounded Phase *mpds*

In this section we study applications of our results to reachability problem among configurations of *mpds*. Reachability easily reduces to parity winning condition.

Definition 9. *Let $\mathcal{H}$ be a* mpds *with Q as its finite set of states. A set $\mathcal{C}$ of configurations of $\mathcal{H}$ is regular if there is a finite multi-automaton which accepts string $s_1\#s_2\#\ldots\#s_l$ starting from state q iff $(q, s_1, s_2, \ldots, s_l) \in \mathcal{C}$.*

The finite multi-automaton in the above definition is just a finite automaton which has one initial state for each $q \in Q$. Finite multi-automata were introduced in [11].

A slightly general version of reachability problem for *mpds* can be defined as the following decision problem.

Definition 10. *(Regular reachability problem) Given an* mpds $\mathcal{M}$ *and a regular set R of configurations of $\mathcal{M}$, is there a $r \in R$ and such that configuration r is reachable from the initial configuration.*

Theorem 3. *Regular reachability problem for bounded phase* mpds *is decidable in time $(|Q|.|\Gamma|)^{l^{O(k+l)}}$, where $|Q|$ is the sum of states in input* mpds *and in input multi-automaton accepting R, Γ is stack alphabet, k is the bound on phases and $l \geq 2$ is the number of stacks in the input* mpds.

Proof. The idea is to add transitions to the input *mpds* to check if its configuration is in R. This can be done in $2l$ phases. Easy details of this proof are given in full version. □

Corollary 1. *Emptiness problem of bounded phase nondeterministic multi-stack pushdown automata is decidable in $(|Q|.|\Gamma|)^{l^{O(k+l)}}$ time where $|Q|$ is the number of states, Γ is stack alphabet, k is bound on phases and $l \geq 2$ is the number of stacks in input* mpda.

Proof. Let given multi-stack pushdown automaton accept by reaching the final state q_f. We convert this *mpda* to *mpds* by erasing input symbols from transitions. Emptiness problem of the given automaton is the same as regular reachability problem of the resulting *mpds* with $R = \{q_f\} \times (\Gamma^*)^l$. □

If we keep fixed all parameters of *mpda* except the number of phases allowed, by the above corollary we get the time complexity as a function of k, to be $2^{2^{O(k)}}$. This is same as the complexity of emptiness checking of an *mpda* in [2].

Consider a multi-stack pushdown system $\mathcal{M}$ with 3 stacks. Define it's transition function so that if stack 1 is empty then it has no transition, otherwise it pops a symbol from stack 1 and pushes a b on stack 2 and a c on stack 3. After this $\mathcal{M}$ again checks stack 1 and repeats the same sequence of actions. If $\mathcal{M}$ starts with initial configuration $(q_0, \bot.a^n, \bot, \bot)$ then it reaches $(q_0, \bot, \bot.b^n, \bot.c^n)$. This shows that $post^*$ of a regular set is not regular. We do not have any example to show that pre^* of a regular set is not regular. In fact, we think that pre^* of a regular set is regular for bounded phase *mpds*.

6 Conclusion

In this paper we have shown that parity games over bounded phase multi-stack pushdown systems can be effectively solved. The complexity of our algorithm is a tower of exponentials of the height same as the number of phases allowed. An open question is that if this complexity can be improved or is there a matching lower bound. In [7,8], winning regions in parity games over pds have been shown to be regular. The same question can also be asked for two player parity games over bounded context switching *mpds* and over bounded phase *mpds*. We think that our techniques can be used to show that winning region in these games is also regular. It will also be interesting to know if MSO theory of configuration graphs of bounded phase *mpds* is decidable.

Finally, we have also mentioned application of our results in deciding emptiness of multi-stack bounded phase $\omega-$automata. It seems interesting to study the class of languages recognized by such $\omega-$automata.

Acknowledgments. Financial support for this work is provided by Research I Foundation.

References

1. Madhusudan, P., Parlato, G., La Torre, S.: A Robust Class of Context-Sensitive Languages. In: 22nd IEEE Symp. on Logic in Computer Science (LICS) Wroclaw, Poland (2007)
2. Madhusudan, P., Parlato, G., La Torre, S.: An Infinite Automaton Characterization of Double Exponential Time. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 33–48. Springer, Heidelberg (2008)
3. Madhusudan, P., Parlato, G., La Torre, S.: Context-Bounded Analysis of Concurrent Queue Systems. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 299–314. Springer, Heidelberg (2008)
4. Lal, A., Reps, T.: Reducing Concurrent Analysis Under a Context Bound to Sequential Analysis. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 37–51. Springer, Heidelberg (2008)
5. Qadeer, S., Rehof, J.: Context-bounded model checking of concurrent software. In: Proceedings of the 11th International Symposium on Tools and Algorithms for the Construction and Analysis of Systems (2005)
6. Kahlon, V., Ivancic, F., Gupta, A.: Reasoning About Threads Communicating via Locks. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 505–518. Springer, Heidelberg (2005)
7. Serre, O.: Note on Winning Positions on Pushdown Games with Omega-Regular Conditions. Information Processing Letters 85(6), 285–291 (2003)
8. Cachat, T.: Uniform solution of parity games on prefix-recognizable graphs. In: Proc. INFINITY. ENTCS, vol. 68(6) (2002)
9. Walukiewicz, I.: Pushdown processes: games and model checking. Information and computation 164, 234–263 (2001)

10. Jurdziński, M.: Small Progress Measures for Solving Parity Games. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 290–301. Springer, Heidelberg (2000)
11. Bouajjani, A., Esparza, J., Maler, O.: Reachability analysis of pushdown automata: Applications to model checking. In: Proc. Concur, pp. 135–150 (1997)
12. Thomas, W.: Languages, automata and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. III, pp. 389–455. Springer, New York (1997)

Data Privacy for $\mathcal{ALC}$ Knowledge Bases

Phiniki Stouppa and Thomas Studer

Universität Bern, Institut für Informatik und angewandte Mathematik,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{stouppa,tstuder}@iam.unibe.ch

Abstract. Information systems support data privacy by granting access only to certain (public) views. The data privacy problem is to decide whether hidden (private) information may be inferred from the public views and some additional general background knowledge. We study the problem of provable privacy in the context of $\mathcal{ALC}$ knowledge bases. First we show that the $\mathcal{ALC}$ privacy problem wrt. concept retrieval and subsumption queries is ExpTime-complete. Then we provide a sufficient condition for data privacy that can be checked in PTime.

1 Introduction

In the context of information systems, the problem of data privacy is to verify whether the *confidential* information that is stored in a system is not provided to unauthorized users and therefore, personal and other sensitive data remain private. Data privacy issues are particularly critical in environments where sharing and reuse of information are constantly applied.

Such an area is, for example, the semantic web. There, knowledge is represented by ontologies which provide formalizations of concept definitions for an application domain. These ontologies are expressed in an ontology language. OWL (Web Ontology Language) is the W3C endorsed standard language for this purpose. The underlying formal framework of OWL are the so-called description logics [1]. In the present paper we will study the privacy problem with respect to the basic description logic $\mathcal{ALC}$ which is the simplest description logic that is boolean closed.

It was always clear that privacy issues have to be considered in the context of ontology languages. Let us cite the OWL Language Guide [2]: '...the capability to merge data from multiple sources, combined with the inferential power of OWL, does have potential for abuse. Users of OWL should be alert to the potential privacy implications.'

The present paper is the continuation of our work started in [3,4]. There, we introduced the problem of *provable data privacy on views* as follows. Assume that some agent has access to a view provided by an information system. Additionally, there is some background knowledge that is publicly available. The privacy problem under this setting is to decide whether the user is not able to infer - from the view and the background knowledge - any answer to a given query q. That one cannot infer any answer to q is formalized as *the set of certain*

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 409–421, 2009.

answers to q is empty. If the problem is answered positively, we say that privacy is *preserved* for q.

We will now use the notion of provable privacy to study a more general problem: namely, the problem of deciding *data privacy on view definitions*. The new problem is now the following: given only a view definition instead of a complete view, decide whether privacy is preserved on all possible views of that view definition. We investigate the new problem for the case of $\mathcal{ALC}$ knowledge bases with general concept inclusion axioms (GCIs). In such a knowledge base the domain is only partially known (incomplete), background knowledge is formalized as a part of the knowledge base, and for the view and the privacy condition we allow for concept retrieval and subsumption queries.

Let us now illustrate the difference between privacy on views and privacy on view definitions. Our running example will be a business information system storing information about account managers and their salaries.

Example 1. The background knowledge states that an account manager gets a high or a low salary:

$$\texttt{account\_manager} = \texttt{high} \sqcup \texttt{low}.$$

Assume that an agent has access to the views defined by

$$\{\texttt{account\_manager}, \neg\texttt{high}\}$$

and that for some reason the extension of `low` should be hidden.

For the *privacy problem on views*, we assume that we are given the answers to the views. For instance, assume $\{a\}$ is the answer of the query `account_manager` and $\{b\}$ is the answer to the query $\neg$`high`. In this case, privacy for `low` is preserved with respect to the given view, since for no individual we can infer that it belongs to `low`.

For the *privacy problem on view definitions*, we do *not* assume that the answers to the views are given. Rather the question is whether privacy is preserved for all possible sets of answers. In our example, privacy is not preserved on the view definition. Consider the following possibility: the answer to the query `account_manager` might be $\{a, b\}$ and the answer to the query $\neg$`high` might be $\{b\}$. In this case b must belong to `low`. Thus privacy is not preserved for `low` with respect to the view definition.

In the next section, we present the syntax and the semantics of $\mathcal{ALC}$, explain how a query is answered on an $\mathcal{ALC}$ knowledge base, and recall from [3,4] the problem of provable data privacy on a given view. Then, in Section 3 we define data privacy on a view definition. We show that in order to decide this problem it is enough to consider a finite number of possible views. As a corollary we obtain that the problem is ExpTime-complete. Moreover, we present a syntactic condition on the knowledge base and the view definition which is sufficient for data privacy. This condition can be checked in PTime. We discuss related work in Section 4. Then we conclude and give some directions for further work.

This paper comes together with a technical report [5]. There we introduce a deductive system for $\mathcal{ALC}$ and apply proof-theoretic techniques in order to give detailed proofs of our results.

2 Preliminaries

The language of $\mathcal{ALC}$ consists of a countable set of *individuals* Ind, a countable set of *atomic concepts* AConc, a countable set of *roles* Rol and the *concepts* built on AConc and Rol as follows:

$$C, D := A \mid \neg A \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

where $A \in \mathsf{AConc}$, $R \in \mathsf{Rol}$, and C and D are concepts. Individuals are denoted by $a, b, c, \ldots$.

Note that the language includes only concepts in negation normal form. The complement of a concept $\neg(C)$ is inductively defined, as usual, by using the law of double negation, de Morgan's laws and the dualities for quantifiers. When the scope of the negation is unambiguous, we also write $\neg C$ instead of $\neg(C)$. Moreover, the constants $\top$ and $\bot$ abbreviate $A \sqcup \neg A$ and $A \sqcap \neg A$, respectively, for some $A \in \mathsf{AConc}$. The set of *subterms* $s(C)$ of a concept C is defined by:

$$\begin{aligned} s(A) &:= \{A\} & s(\neg A) &:= \{\neg A\} \\ s(C \star D) &:= \{C \star D\} \cup s(C) \cup s(D) & s(QR.C) &:= \{QR.C\} \cup s(C) \end{aligned}$$

where $\star$ is either $\sqcup$ or $\sqcap$ and Q is either $\forall$ or $\exists$. Note that the complements of atomic concepts are not decomposable. That means, for instance, the subterms of $A_1 \sqcup \exists R.\neg A_2$ are $A_1, \neg A_2, \exists R.\neg A_2$ and $A_1 \sqcup \exists R.\neg A_2$.

Concepts are interpreted in the usual way:

Definition 1. *An interpretation $\mathcal{I}$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and a mapping $(\cdot)^{\mathcal{I}}$ that assigns*

- *to each individual $a \in \mathsf{Ind}$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*
- *to each atomic concept $A \in \mathsf{AConc}$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$*
- *to each role $R \in \mathsf{Rol}$ a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*

The elements of a domain are denoted by $d, d_1, d_2, \ldots$. The interpretation $\mathcal{I}$ extends then on concepts as follows:

$$\begin{aligned} (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} &= \{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 \ ((d_1, d_2) \in R^{\mathcal{I}} \Rightarrow d_2 \in C^{\mathcal{I}})\} \\ (\exists R.C)^{\mathcal{I}} &= \{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2 \ ((d_1, d_2) \in R^{\mathcal{I}} \ \& \ d_2 \in C^{\mathcal{I}})\} \end{aligned}$$

We can now define the notion of a knowledge base and its models. An $\mathcal{ALC}$ knowledge base $\mathcal{O}$ is the union of

1. a finite *terminological* set (TBox) of *inclusion axioms* that have the form $\top \sqsubseteq C$,[1] where C is called *inclusion concept*, and
2. a finite *assertional* set (ABox) of assertions of the form $a : C$ (*concept assertion*) or $(a, b) : R$ (*role assertion*) where R is called *assertional role* and C is called *assertional concept*.

We denote the set of individuals that appear in $\mathcal{O}$ by $\mathsf{Ind}(\mathcal{O})$. An interpretation $\mathcal{I}$ is a *model* of

- an inclusion axiom $\top \sqsubseteq C$ ($\mathcal{I} \models \top \sqsubseteq C$) if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$,
- a concept assertion $a : C$ ($\mathcal{I} \models a : C$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
- a role assertion $(a, b) : R$ ($\mathcal{I} \models (a, b) : R$) if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

Let $\mathcal{O}$ be the $\mathcal{ALC}$-knowledge base of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. An interpretation $\mathcal{I}$ is a model of $\mathcal{O}$ if $\mathcal{I} \models \phi$, for every $\phi \in \mathcal{T} \cup \mathcal{A}$. A knowledge base $\mathcal{O}$ is *consistent* if it has a model. Moreover, for ψ an inclusion axiom or an assertion, we say that $\mathcal{O} \models \psi$ (in words, $\mathcal{O}$ *entails* ψ) if for every model $\mathcal{I}$ of $\mathcal{O}$, $\mathcal{I} \models \psi$ also holds.

The consistency problem for $\mathcal{ALC}$ is ExpTime-complete, see for instance [1]. The entailment problem is reducible to the consistency problem as follows:

Theorem 1. *Let $\mathcal{O}$ be an $\mathcal{ALC}$ knowledge base and n_{ew} be an individual not belonging to $\mathsf{Ind}(\mathcal{O})$. Then,*

- $\mathcal{O} \models \top \sqsubseteq C$ *iff* $\mathcal{O} \cup \{n_{ew} : \neg C\}$ *is inconsistent and*
- $\mathcal{O} \models a : C$ *iff* $\mathcal{O} \cup \{a : \neg C\}$ *is inconsistent.*

Theorem 1 shows that an entailment can be decided in ExpTime. Moreover, the inconsistency problem is reducible to the entailment problem and so, deciding an entailment is an ExpTime-complete problem, too.

The reasoning tasks on an $\mathcal{ALC}$ knowledge base are formulated below as *queries*. For the time being we consider only subsumption and retrieval queries.

Definition 2. *An $\mathcal{ALC}$* query *q is either a concept of $\mathcal{ALC}$ (called retrieval query) or an inclusion axiom (called boolean query). The* answer to a query *q with respect to an $\mathcal{ALC}$ knowledge base $\mathcal{O}$ ($\mathsf{ans}(q, \mathcal{O})$) is given as follows where tt is a special constant denoting 'true'.*

$$\begin{aligned} \mathsf{ans}(\top \sqsubseteq C, \mathcal{O}) &:= \{\mathsf{tt}\} \text{ , if } \mathcal{O} \models \top \sqsubseteq C, \\ \mathsf{ans}(\top \sqsubseteq C, \mathcal{O}) &:= \emptyset \text{ , if } \mathcal{O} \not\models \top \sqsubseteq C, \\ \mathsf{ans}(C, \mathcal{O}) &:= \{a \in \mathsf{Ind}(\mathcal{O}) \mid \mathcal{O} \models a : C\} \ . \end{aligned}$$

A view definition *V is a finite set of $\mathcal{ALC}$ queries.*

Definition 3. *A* view *V_I of a view definition V is a total function with domain V such that if $\langle q, r \rangle \in V_I$, then*

[1] This form does not restrict a knowledge base since an arbitrary inclusion $C_1 \sqsubseteq C_2$ can be linearly transformed to its equivalent $\top \sqsubseteq \neg C_1 \sqcup C_2$.

1. $r \subseteq \mathsf{Ind}$ *and finite if* q *is a retrieval query,*
2. $r \subseteq \{tt\}$ *if* q *is a boolean query.*

We say, that an $\mathcal{ALC}$ *knowledge base* $\mathcal{O}$ entails *a view* V_I *(*$\mathcal{O} \models V_I$*) if for each* $\langle q, r\rangle \in V_I$ *we have* $r = ans(q, \mathcal{O})$.

Note that a view can also be formulated as a set A_{V_I} of axioms and assertions. We set

$$A_{V_I} := \{\top \sqsubseteq C \mid \langle \top \sqsubseteq C, \{\mathsf{tt}\}\rangle \in V_I\} \cup \\ \{a : C \mid \text{there is a set } \mathsf{In} \text{ with } \langle C, \mathsf{In}\rangle \in V_I \text{ and } a \in \mathsf{In}\}.$$

Our notion of a view entailed by a knowledge base relates to the standard notion of entailment as follows. Let V_I be a view of a view definition V such that $\mathcal{O} \models V_I$ for some $\mathcal{O}$. For each retrieval query C in V and all individuals a we have $C(a) \in A_{V_I}$ iff $\mathcal{O} \models C(a)$. For each boolean query $\top \sqsubseteq C$ in V we have $\top \sqsubseteq C \in A_{V_I}$ iff $\mathcal{O} \models \top \sqsubseteq C$.

We turn now to the problem of provable data privacy wrt. views. This problem has been examined for arbitrary data and knowledge bases in [3,4]. Here we present the problem from the point of view of $\mathcal{ALC}$ knowledge bases and queries; we additionally admit that the underlying knowledge base is always consistent.

The problem assumes that a user is granted access to a specific view V_I and to some general (background) knowledge of such a knowledge base. In our case we assume that all information about the knowledge base is stated explicitly in it and, therefore, the background knowledge coincides with a part of the knowledge base. We call this knowledge base $\mathcal{O}_{bg}$.

Informally, we say that *data privacy is preserved* for a query q with respect to $\langle \mathcal{O}_{bg}, V_I\rangle$ if there are no answers to q that follow with certainty from the information of V_I and $\mathcal{O}_{bg}$. This can be made precise by the notion of certain answer. The function $\mathsf{certain}(q, \langle \mathcal{O}_{bg}, V_I\rangle)$ returns the answers to q that hold in every knowledge base that - according to the user's knowledge - could be the actual one (a so-called *possible* knowledge base).

Definition 4. *A knowledge base* $\mathcal{P}$ *is* possible *wrt.* $\langle \mathcal{O}_{bg}, V_I\rangle$ *if* $\mathcal{P}$ *is consistent,* $\mathcal{O}_{bg} \subseteq \mathcal{P}$*, and* $\mathcal{P} \models V_I$*. By* $\mathsf{Poss}_{\langle \mathcal{O}_{bg}, V_I\rangle}$*, we denote the set of all possible knowledge bases with respect to* $\langle \mathcal{O}_{bg}, V_I\rangle$*.*

In the sequel we consider only $\langle \mathcal{O}_{bg}, V_I\rangle$ tuples with $\mathsf{Poss}_{\langle \mathcal{O}_{bg}, V_I\rangle} \neq \emptyset$ which means that $\mathcal{O} \cup A_{V_I}$ is satisfiable.

Definition 5. *The* certain answers *to a query* q *wrt.* $\langle \mathcal{O}_{bg}, V_I\rangle$ *are defined by*

$$\mathsf{certain}(q, \langle \mathcal{O}_{bg}, V_I\rangle) := \bigcap_{\mathcal{P} \in \mathsf{Poss}_{\langle \mathcal{O}_{bg}, V_I\rangle}} \mathsf{ans}(q, \mathcal{P}).$$

Definition 6. *Given a knowledge base* $\mathcal{O}_{bg}$*, a view* V_I *and a query* q*,* data privacy is preserved for q with respect to $\langle \mathcal{O}_{bg}, V_I\rangle$ *if*

$$\mathsf{certain}(q, \langle \mathcal{O}_{bg}, V_I\rangle) = \emptyset.$$

Note that this privacy notion is based on positive answers only. So we may have privacy for a query C even when we know with certainty that some individual a does not belong to C. The extreme case is when $\top \sqsubset \neg C$ is public knowledge. Then we know that C must be empty and still we have privacy for C (since there is no element for which we can infer that it belongs to C).

There are situations in which the certain answers to a query q can be computed by issuing q against a particular fixed data or knowledge base, see for instance [6,3]. In our setting, we simply can take $\mathcal{O}_{bg} \cup A_{V_I}$ for this purpose. Namely, we have

$$\mathsf{certain}(q, \langle \mathcal{O}_{bg}, V_I \rangle) = \mathsf{ans}(q, \mathcal{O}_{bg} \cup A_{V_I}).$$

Therefore, we immediately get the following result.

Theorem 2 (see [4, Corollary 1]). *Data privacy is preserved for a query q wrt. a view V_I and a knowledge base $\mathcal{O}_{bg}$ if and only if*

$$\mathsf{ans}(q, \mathcal{O}_{bg} \cup A_{V_I}) = \emptyset.$$

According to Definition 2, $\mathsf{ans}(q, \mathcal{O}_{bg} \cup A_{V_I})$ can be computed by a number of entailments which is linear the size of $\mathcal{O}_{bg} \cup A_{V_I}$. If q is a retrieval query, then we need one entailment check for each individual occurring in $\mathcal{O}_{bg} \cup A_{V_I}$. If q is a boolean query, then we trivially need only one entailment check. As it has already been stated, the entailment problem is reducible to the consistency problem which is solvable in ExpTime. Therefore, Theorem 2 provides an ExpTime decision procedure for the problem of data privacy on views.

The problem of $\mathcal{ALC}$ concept satisfiability wrt. a consistent TBox is also ExpTime-hard, see [7] and [1]. Note that the proof in [1] does not necessarily construct a *consistent* TBox, however an easy modification will do the job. Therefore, the $\mathcal{ALC}$ data privacy problem is ExpTime-complete since we have that a concept C is unsatisfiable wrt. a TBox $\mathcal{T}$ iff data privacy for $\top \sqsubseteq \neg C$ wrt. $\mathcal{T}$ and the empty view is not preserved.

Corollary 1. *The problem of $\mathcal{ALC}$ data privacy for a query wrt. a view and a knowledge base is ExpTime-complete.*

3 Data Privacy on View Definitions

Let us now introduce the problem of provable data privacy wrt. view definitions. First, we introduce the following auxiliary notion.

Definition 7. *Let $\mathcal{O}_{bg}$ be an $\mathcal{ALC}$ knowledge base and V be a view definition. A view V_I* is based on *$\langle \mathcal{O}_{bg}, V \rangle$ if it satisfies the following: (i) V_I is a view of V and (ii) $\mathsf{Poss}_{\langle \mathcal{O}_{bg}, V_I \rangle} \neq \emptyset$.*

Definition 8. *Let $\mathcal{O}_{bg}$ be an $\mathcal{ALC}$ knowledge base and V be a view definition. Data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V \rangle$ if for every view V_I based on $\langle \mathcal{O}_{bg}, V \rangle$, data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V_I \rangle$. The data privacy problem on view definitions is to decide whether data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V \rangle$.*

Example 2. We consider again the business information system storing information about key accounts, account managers, and their salaries. The general background knowledge states the following: An account manager gets a high or a low salary, see (1). If someone gets a high salary, then she handles key accounts only, see (2). The domain of the `handles` relation is the set of account managers, see (3). Formally, $\mathcal{O}_{bg}$ is the set of the following axioms:

$$\texttt{account\_manager} = \texttt{high} \sqcup \texttt{low} \tag{1}$$

$$\texttt{high} \sqsubseteq \forall \texttt{handles}.\texttt{key\_account} \tag{2}$$

$$\exists \texttt{handles}.\top = \texttt{account\_manager} \tag{3}$$

Consider the view definition $V_1 := \{\exists \texttt{handles}.\texttt{key\_account}\}$. Given this setting, the following two statements, for example, hold:

$$\text{privacy is preserved for } \texttt{key\_account} \text{ with respect to } \langle \mathcal{O}_{bg}, V_1 \rangle \tag{$*$}$$

$$\text{privacy is preserved for } \texttt{low} \text{ with respect to } \langle \mathcal{O}_{bg}, V_1 \rangle. \tag{$**$}$$

That means an agent who is granted access to the view provided by V_1 cannot infer which are the key accounts nor who gets a low salary.

To see ($*$), simply observe that the only information we obtain from a non-empty answer to the query in V_1 is that the extension of `key_account` cannot be empty. However, we do not get any knowledge about which individual belongs to it. There is a possible knowledge base in which only some individual a belongs to `key_account` and there is another possible knowledge base in which only some other individual b belongs to `key_account`. Therefore the set of certain answers to `key_account` is empty and thus privacy is preserved.

To see ($**$), simply observe that we always can choose `low` to be the empty concept, no matter what the answer to the query in V_1 is.

Consider now the view definition

$$V_2 := \{\exists \texttt{handles}.\neg\texttt{key\_account}\}.$$

Privacy is not preserved for `low` with respect to $\langle \mathcal{O}_{bg}, V_2 \rangle$. This can be seen as follows. Assume that issuing the view query against some knowledge base $\mathcal{KB}$ gives

$$a \in \mathsf{ans}(\exists \texttt{handles}.\neg\texttt{key\_account}, \mathcal{KB}),$$

for some individual a. By (2), we get $\mathcal{KB} \models a : \neg\texttt{high}$ and by (3) we find $\mathcal{KB} \models a : \texttt{account\_manager}$. Thus (1) yields $\mathcal{KB} \models a : \texttt{low}$. We conclude that privacy is not preserved for `low`.

The problem of data privacy on a view definition is decidable since it is enough to consider only the views entailed by a finite set of knowledge bases $\mathbb{P}$. Given $\langle \mathcal{O}_{bg}, V \rangle$ and an individual $n_{ew} \notin \mathsf{Ind}(\mathcal{O}_{bg})$, a knowledge base P is *possible* if

1. $P \supseteq \mathcal{O}_{bg}$ and consistent,

2. if $\top \sqsubseteq C \in P$ then $\top \sqsubseteq C \in \mathcal{O}_{bg} \cup V$, and
3. if $a : C \in P$ then $a : C \in \mathcal{O}_{bg}$ or $(a \in \mathsf{Ind}(\mathcal{O}_{bg}) \cup \{n_{ew}\}$ and $C \in V)$.

Then $\mathbb{P}$ is the set of all possible P wrt. $\langle \mathcal{O}_{bg}, V \rangle$ and n_{ew}.

Theorem 3. *Let $\mathcal{O}$ be an $\mathcal{ALC}$ knowledge base and V be a view definition. Data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V \rangle$ if and only if, for every view V_I of V that is entailed by some $P \in \mathbb{P}$, data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V_I \rangle$.*

A proof is presented in [5]. A naive ExpTime decision procedure for this problem can be constructed directly from the above theorem: first compute $\mathbb{P}$ and all views entailed by its knowledge bases, and then decide data privacy on each of these views. Let P^+ be the knowledge base constructed from $\mathcal{O}_{bg}$ and V as follows:

$$P^+ = \{\top \sqsubseteq C \in V\} \cup \bigcup \{a : C \mid (a \in \mathsf{Ind}(\mathcal{O}_{bg}) \cup \{n_{ew}\}) \text{ and } C \in V\}.$$

Then, $\mathbb{P}$ can be constructed by first computing all subsets of P^+ and then checking their consistency wrt. $\mathcal{O}_{bg}$. Since P^+ can be constructed polynomially wrt. the size of $\mathcal{O}_{bg}$ and V, there are at most $2^{p(n)}$ subsets of P^+ of maximal cardinality $p(n)$, where n is the total size of $\mathcal{O}_{bg}, V$ and q. Since consistency is decidable in ExpTime, computing $\mathbb{P}$ stays in ExpTime. Now, in order to compute the views entailed by some $P \in \mathbb{P}$, a polynomial number of entailments on every $P \in \mathbb{P}$ is required. Therefore the computation of all views stays also in ExpTime. Finally, Corollary 1 together with the fact that V_I grows polynomially wrt. the size of V and P, imply that the total time required for checking privacy on all of the (at most) exponentially many views is again exponential wrt. n.

The problem of data privacy on view definitions is also ExpTime-hard as the corresponding problem on views is polynomially reducible to this problem: data privacy for q is preserved wrt. $\mathcal{O}_{bg}$ and V_I iff it is preserved wrt. $\mathcal{O}_{bg} \cup A_{V_I}$ and the empty view definition.

Theorem 4. *The problem of $\mathcal{ALC}$ data privacy on view definitions is ExpTime-complete.*

In the sequel we present a condition on $\mathcal{O}_{bg}, V$ and q which can be decided in PTime and implies data privacy for q wrt. $\langle \mathcal{O}_{bg}, V \rangle$. Thus, we have a sufficient condition for data privacy that can be checked efficiently. It is based on the syntactic structure of the concepts that constitute the background knowledge and the view definition. We begin by excluding some 'common sense' queries from being potential secrets, because of their trivial (partial) answers.

Definition 9. *A* query q is trivial wrt. a tuple $\langle \mathcal{O}_{bg}, V \rangle$ *when*

- $ans(q, \emptyset) = \{tt\}$ *(i.e. $\emptyset \models q$) and q is a boolean query*
- $ans(\top \sqsubseteq q, \emptyset) = \{tt\}$*, q is a retrieval query C, and in addition* $\mathsf{Ind}(\mathcal{O}_{bg}) = \emptyset$ *implies* $\exists C \in V(\mathcal{O}_{bg} \not\models \top \sqsubseteq \neg C)$.

A retrieval query might violate privacy only if some individuals are (potentially) given in public. This is the reason for the condition posed on retrieval queries in

the above definition. An $\mathcal{ALC}$ query qualifies as a *privacy condition on a tuple* $\langle \mathcal{O}_{bg}, V \rangle$ if it is not trivial wrt. $\langle \mathcal{O}_{bg}, V \rangle$.

Next, we define the boolean function $s_{afe}()$ that decides whether a concept D or a role R exhibits some information about q. Given a knowledge base $\mathcal{O}_{bg}$, a view definition V and a privacy condition q on $\langle \mathcal{O}_{bg}, V \rangle$, the information about a concept D is *safe* if $s_{afe}(D, q)$ returns 1; and the information of a role R is safe if $s_{afe}(R, \langle \mathcal{O}_{bg}, V, q \rangle)$ returns 1.

In the sequel, we use the following conventions. *Concepts and roles of a tuple* $\langle \mathcal{O}_{bg}, V \rangle$ are all inclusion and assertional concepts, assertional roles, and retrieval queries that appear in $\mathcal{O}_{bg}$ or V. If a concept C_2 has a subterm C_1 then C_2 is also written as $C_2[C_1]$. If, in addition, there is an occurrence of C_1 in C_2 that is not prefixed with a quantifier, then C_2 may also be written as $C_2[C_1]^0$. Similarly, if we want to emphasize that C_1 is not prefixed in C_2 with an existential quantifier, then C_2 may also be written as $C_2[C_1]^{0^\exists}$. For example, the concept $A_1 \sqcup \forall R_2.\neg A_2$ can be also written as $A_1 \sqcup \forall R_2.\neg A_2[\neg A_2]$ or as $A_1 \sqcup \forall R_2.\neg A_2[\neg A_2]^{0^\exists}$ but not as $A_1 \sqcup \forall R_2.\neg A_2[\neg A_2]^0$.

Now, assume we are given a query q^{c} where C is the inclusion or assertional concept of q (i.e. $q^{\mathrm{c}} = \top \sqsubseteq C$ or $q^{\mathrm{c}} = C$). The function $s_{afe}()$ is defined on concepts and roles as follows:

For a concept D, $s_{afe}(D, q^{\mathrm{c}}) = 1$ iff there are no D_1 and C_1 subterms of D and C, respectively, of the form:

a. $D_1 = C_1 = A$, or
b. $D_1 = C_1 = \neg A$, or
c. $D_1 = QR.D_2$ and $C_1 = QR.C_2$,

where $A \in \mathsf{AConc}$, $R \in \mathsf{Rol}$ and $Q \in \{\forall, \exists\}$, and for which either

1. $D[D_1]^0$ and $C[C_1]^{0^\exists}$ hold, or
2. $D[D_1]^0$, $C[\exists R.C'[C_1]]^{0^\exists}$ and $C[\forall R.C'']$ hold.

For a role R and a tuple $\langle \mathcal{O}_{bg}, V \rangle$, $s_{afe}(R, \langle \mathcal{O}_{bg}, V, q^{\mathrm{c}} \rangle) = 1$ iff:

1. C is not of the form $C[\exists R.C']^0$ and
2. for every concept D_2 for which there is a concept $D_1[\forall R.D_2]^{0^\exists}$ of $\langle \mathcal{O}_{bg}, V \rangle$, $s_{afe}(D_2, q^{\mathrm{c}}) = 1$.

The following theorem provides a sufficient condition for privacy on view definition. A proof can be found in our technical report [5]. There, the theorem is established by proof-theoretic investigations of a sequent system for $\mathcal{ALC}$.

Theorem 5. *Given a consistent $\mathcal{ALC}$ knowledge base $\mathcal{O}_{bg}$, a view definition V and a privacy condition q on $\langle \mathcal{O}_{bg}, V \rangle$, data privacy is preserved for q wrt. $\langle \mathcal{O}_{bg}, V \rangle$ if for every concept D and role R of $\langle \mathcal{O}_{bg}, V \rangle$*

$$s_{afe}(D, q) = s_{afe}(R, \langle \mathcal{O}_{bg}, V, q \rangle) = 1.$$

Moreover, it can be decided in PTime whether for every concept D and role R of $\langle \mathcal{O}_{bg}, V \rangle$ we have $s_{afe}(D, q) = s_{afe}(R, \langle \mathcal{O}_{bg}, V, q \rangle) = 1$.

Example 3. Consider again the setting of Example 2. We can establish $(*)$ by the previous theorem. Let q be the query `key_account`. We have

$$s_{afe}(\texttt{account\_manager}, q) = s_{afe}(\texttt{high} \sqcup \texttt{low}, q) = s_{afe}(\texttt{high}, q)$$
$$= s_{afe}(\exists \texttt{handles}.\top, q) = 1.$$

We also have

$$s_{afe}(\forall \texttt{handles}.\texttt{key\_account}, q) = 1$$
$$s_{afe}(\exists \texttt{handles}.\texttt{key\_account}, q) = 1$$

since `key_account` occurs only behind a quantifier in the concepts of $\mathcal{O}_{bg}$ and V_1. Therefore the condition of Theorem 5 is satisfied and thus privacy is preserved for `key_account`.

However, Theorem 5 does not yield $(**)$ since `low` is an atomic subterm of $\texttt{high} \sqcup \texttt{low}$ which is not behind a quantifier.

4 Related Work

The notion of certain answer originates from the study of incomplete databases [8] and is now a key notion in data integration [9,10] and data exchange [11,6]. Obviously, our work on privacy in ontologies is tightly related to privacy in incomplete databases which has been studied by several authors.

Nash and Deutsch [12], for instance, study privacy for database integration. Like us, they are interested in logical security. That is all an attacker can do is issue queries and apply arbitrary computational power on the answers to these queries together with background knowledge to obtain the secret. They introduce several notions of privacy that are suitable for a data integration scenario and study the corresponding algorithms.

Another approach is to preserve confidentially at runtime. At each query, it is checked whether the answer would leak hidden information. If this is the case, then the answer is distorted. Biskup and Weibert [13] adopt this approach for incomplete databases. In their setting a database is simply a set of propositional sentences and queries simply return *yes, no,* or *undef.* They investigate several distortion methods (lying, refusal and a combination thereof) which guarantee that a user cannot learn classified information.

Our notion of provable data privacy only guarantees that, given a concept C, for no individual a we can infer $a : C$. Example 2 shows that $(*)$ holds even if we know that `key_account` cannot be empty. *Perfect privacy* is a much more restrictive notion than provable privacy. It guarantees that an answer to a query does not change the attacker's a priori belief about the secret. This belief is modeled as a probability distribution with the assumption that the tuples in the secret answer are independent events. Perfect privacy has been introduced in [14] and generalized in [15]. Recently, a connection between perfect privacy and query containment has been established [16] which allows to identify subclasses of conjunctive queries for which enforcing perfect privacy is tractable. Dalvi et al. [17]

argue that perfect privacy is often too restrictive for practical applications. They provide a new probabilistic database model for practical privacy and study five privacy characterizations for it, including perfect privacy and certain answers.

It is necessary to consider an attacker's background knowledge when reasoning about privacy. We have chosen a simple approach: we fix the background knowledge and model it as a part of the knowledge base. The general case is when the background knowledge is not given in advance. Recently, a formal study of this so-called worst-case background knowledge has been initiated [18].

The problem of privacy aware access to ontologies is also addressed in [19]. There it is shown how view based query answering is able to conceal from the user information that are not logical consequences of the associated authorization views. The authors introduce several different semantics for view based query answering which in turn conceal different amounts of information when applied to the privacy problem. The semantics which corresponds to our approach is called TBox-centered semantics. There the user is aware of the TBox which in our setting is expressed by the TBox being part of the general background knowledge.

Grau and Horrocks [20] study different privacy guarantees for logic-based information systems. They present privacy preserving query answering as reasoning problems and establish a general connection between such reasoning problems and probabilistic privacy guarantees. The reasoning problems they introduce are related to certain notions of conservative extension which occur in the context of modular ontologies.

5 Conclusions

We have studied the problem of provable data privacy on view definitions for $\mathcal{ALC}$ knowledge bases. Our goal was to verify that a given privacy condition holds on all possible views of a given definition. We have presented an ExpTime-complete decision procedure for this privacy problem. Moreover, we have studied a syntactic condition which is sufficient for provable privacy and which can be decided in PTime.

Our work is preliminary in the sense that we treat only the case of $\mathcal{ALC}$ knowledge bases and views that are simple queries. There are two important generalizations of our results which will be addressed in future work.

First we have only considered $\mathcal{ALC}$ knowledge bases. $\mathcal{ALC}$ is the basic description logic language and therefore a natural candidate for an initial study. However, current ontology languages are based on very expressive description logics. Future work has to deal with, for instance, $\mathcal{SHOIN}(D)$ [21] which corresponds to OWL DL.

Second we restricted ourselves to concept retrieval and subsumption queries. In this setting, Theorem 3 becomes a consequence of the tree model property. Things are more complex if we also allow role expressions in a view. In a general setting, one also has to consider (union) conjunctive queries over description logics [22]. Then we cannot encode views as knowledge bases, and computing certain answers becomes more difficult.

Acknowledgments

We would like to thank the anonymous referees for the detailed comments which helped to improve the quality of this paper.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
2. Smith, M.K., Welty, C., McGuinness, D.L.: OWL web ontology language guide (2004), http://www.w3.org/TR/owl-guide/
3. Stoffel, K., Studer, T.: Provable data privacy. In: Viborg, K., Debenham, J., Wagner, R. (eds.) DEXA 2005. LNCS, vol. 3588, pp. 324–332. Springer, Heidelberg (2005)
4. Stouppa, P., Studer, T.: A formal model of data privacy. In: Virbitskaite, I., Voronkov, A. (eds.) PSI 2006. LNCS, vol. 4378, pp. 401–411. Springer, Heidelberg (2007)
5. Stouppa, P., Studer, T.: Data privacy for $\mathcal{ALC}$, Technical Report, IAM 08-002 (2008), http://www.iam.unibe.ch/publikationen/techreports/2008/
6. Fagin, R., Kolaitis, P.G., Miller, R., Popa, L.: Data exchange: Semantics and query answering. Theoretical Computer Science 336, 89–124 (2005)
7. Hofmann, M.: Proof-theoretic approach to description-logic. In: LICS 2005, pp. 229–237. IEEE Computer Society, Los Alamitos (2005)
8. van der Meyden, R.: Logical approaches to incomplete information: a survey. In: Logics for databases and information systems, pp. 307–356. Kluwer, Dordrecht (1998)
9. Calì, A., Calvanese, D., Giacomo, G.D., Lenzerini, M.: Data integration under integrity constraints. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 262–279. Springer, Heidelberg (2002)
10. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10(4), 270–294 (2001)
11. Arenas, M., Libkin, L.: XML data exchange: Consistency and query answering. In: PODS, pp. 13–24 (2005)
12. Nash, A., Deutsch, A.: Privacy in glav information integration. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 89–103. Springer, Heidelberg (2006)
13. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Journal of Information Security 7(3), 199–217 (2008)
14. Miklau, G., Suciu, D.: A formal analysis of information disclosure in data exchange. In: SIGMOD, pp. 575–586. ACM, New York (2004)
15. Deutsch, A., Papakonstantinou, Y.: Privacy in database publishing. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 230–245. Springer, Heidelberg (2004)
16. Machanavajjhala, A., Gehrke, J.: On the efficiency of checking perfect privacy. In: PODS 2006, pp. 163–172. ACM Press, New York (2006)
17. Dalvi, N.N., Miklau, G., Suciu, D.: Asymptotic conditional probabilities for conjunctive queries. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 289–305. Springer, Heidelberg (2004)

18. Martin, D.J., Kifer, D., Machanavajjhala, A., Gehrke, J., Halpern, J.Y.: Worst-case background knowledge in privacy. In: ICDE, pp. 126–135. IEEE, Los Alamitos (2007)
19. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: View-based query answering over description logic ontologies. In: Principles of Knowledge Representation and Reasoning, pp. 242–251 (2008)
20. Grau, B.C., Horrocks, I.: Privacy-preserving query answering in logic-based information systems. In: 18th European Conference on Artificial Intelligence (ECAI 2008) (to appear, 2008)
21. Horrocks, I., Patel-Schneider, P., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics 1(1) (2003)
22. Calvanese, D., Lenzerini, M.: Answering queries using views over description logics knowledge bases. In: Proceedings of AAAI 2000, pp. 386–391 (2000)

Fixed Point Theorems on Partial Randomness

Kohtaro Tadaki

Research and Development Initiative, Chuo University
1–13–27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
tadaki@kc.chuo-u.ac.jp

Abstract. In our former work [K. Tadaki, Local Proceedings of CiE 2008, pp. 425–434, 2008], we developed a statistical mechanical interpretation of algorithmic information theory by introducing the notion of thermodynamic quantities, such as free energy $F(T)$, energy $E(T)$, and statistical mechanical entropy $S(T)$, into the theory. We then discovered that, in the interpretation, the temperature T equals to the partial randomness of the values of all these thermodynamic quantities, where the notion of partial randomness is a stronger representation of the compression rate by program-size complexity. Furthermore, we showed that this situation holds for the temperature itself as a thermodynamic quantity. Namely, the computability of the value of partition function $Z(T)$ gives a sufficient condition for $T \in (0,1)$ to be a fixed point on partial randomness. In this paper, we show that the computability of each of all the thermodynamic quantities above gives the sufficient condition also. Moreover, we show that the computability of $F(T)$ gives completely different fixed points from the computability of $Z(T)$.

Keywords: Algorithmic randomness, fixed point theorem, partial randomness, Chaitin's Ω number, algorithmic information theory, thermodynamic quantities.

1 Introduction

Algorithmic information theory (AIT, for short) is a framework for applying information-theoretic and probabilistic ideas to recursive function theory. One of the primary concepts of AIT is the *program-size complexity* (or *Kolmogorov complexity*) $H(s)$ of a finite binary string s, which is defined as the length of the shortest binary program for the universal self-delimiting Turing machine U to output s. By the definition, $H(s)$ is thought to represent the degree of randomness of a finite binary string s. In particular, the notion of program-size complexity plays a crucial role in characterizing the *randomness* of an infinite binary string, or equivalently, a real number.

In [14] we developed a statistical mechanical interpretation of AIT. Especially, in the development we introduced the notion of *thermodynamic quantities*, such as partition function $Z(T)$, free energy $F(T)$, energy $E(T)$, statistical mechanical entropy $S(T)$, and specific heat $C(T)$, into AIT. These quantities are real numbers which depend only on temperature T, any positive real number. We then

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 422–440, 2009.

proved that if the temperature T is a computable real number with $0 < T < 1$ then, for each of these thermodynamic quantities, the partial randomness of its value equals to T, where the notion of *partial randomness* is a stronger representation of the compression rate by means of program-size complexity. Thus, the temperature T plays a role as the partial randomness of all the thermodynamic quantities in the statistical mechanical interpretation of AIT. In [14] we further showed that the temperature T plays a role as the partial randomness of the temperature T itself, which is a thermodynamic quantity of itself. Namely, we proved *the fixed point theorem on partial randomness*,[1] which states that, for every $T \in (0,1)$, if the value of partition function $Z(T)$ at temperature T is a computable real number, then the partial randomness of T equals to T, and therefore the compression rate of T equals to T, i.e., $\lim_{n\to\infty} H(T_n)/n = T$, where T_n is the first n bits of the base-two expansion of T.

In this paper, we show that a fixed point theorem of the same form as for $Z(T)$ holds also for each of free energy $F(T)$, energy $E(T)$, and statistical mechanical entropy $S(T)$. Moreover, based on the statistical mechanical relation $F(T) = -T\log_2 Z(T)$, we show that the computability of $F(T)$ gives completely different fixed points from the computability of $Z(T)$.

The paper is organized as follows. We begin in Section 2 with some preliminaries to AIT and partial randomness. In Section 3, we review the previous results [14] on the statistical mechanical interpretation of AIT and the fixed point theorem by $Z(T)$, which is given as Theorem 3 in the present paper. Our main results; the fixed point theorems by $F(T)$, $E(T)$, and $S(T)$, are presented in Section 4, and their proofs are completed in Section 5. In the last section, we investigate some properties of the sufficient conditions for T to be a fixed point in the fixed point theorems.

2 Preliminaries

2.1 Basic Notation

We start with some notation about numbers and strings which will be used in this paper. $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ is the set of natural numbers, and $\mathbb{N}^+$ is the set of positive integers. $\mathbb{Q}$ is the set of rational numbers, and $\mathbb{R}$ is the set of real numbers. Let $f\colon S \to \mathbb{R}$ with $S \subset \mathbb{R}$. We say that f is *increasing* (resp., *non-decreasing*) if $f(x) < f(y)$ (resp., $f(x) \leq f(y)$) for all $x, y \in S$ with $x < y$. We denote by f' the derived function of f.

Normally, $o(n)$ denotes any function $f\colon \mathbb{N}^+ \to \mathbb{R}$ such that $\lim_{n\to\infty} f(n)/n = 0$. On the other hand, $O(1)$ denotes any function $g\colon \mathbb{N}^+ \to \mathbb{R}$ such that there is $C \in \mathbb{R}$ with the property that $|g(n)| \leq C$ for all $n \in \mathbb{N}^+$.

$\{0,1\}^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, \dots\}$ is the set of finite binary strings, where λ denotes the *empty string*. For any $s \in \{0,1\}^*$, $|s|$ is the *length* of s. A subset S of $\{0,1\}^*$ is called *prefix-free* if no string in S is a prefix of another

[1] The fixed point theorem on partial randomness is called a fixed point theorem on compression rate in [14].

string in S. For any partial function f, the domain of definition of f is denoted by $\mathrm{dom}\, f$. We write "r.e." instead of "recursively enumerable."

Let α be an arbitrary real number. $\lfloor \alpha \rfloor$ is the greatest integer less than or equal to α, and $\lceil \alpha \rceil$ is the smallest integer greater than or equal to α. For any $n \in \mathbb{N}^+$, we denote by $\alpha_n \in \{0,1\}^*$ the first n bits of the base-two expansion of $\alpha - \lfloor \alpha \rfloor$ with infinitely many zeros. For example, in the case of $\alpha = 5/8$, $\alpha_6 = 101000$.

We say that a real number α is *computable* if there exists a total recursive function $f\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $|\alpha - f(n)| < 1/n$ for all $n \in \mathbb{N}^+$. We say that α is *left-computable* if there exists a total recursive function $g\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $g(n) \leq \alpha$ for all $n \in \mathbb{N}^+$ and $\lim_{n\to\infty} g(n) = \alpha$. On the other hand, we say that a real number α is *right-computable* if $-\alpha$ is left-computable. The following (i) and (ii) then hold:

(i) A real number α is computable if and only if α is both left-computable and right-computable.
(ii) A real number α is right-computable if and only if the set $\{\, r \in \mathbb{Q} \mid \alpha < r \,\}$ is r.e.

See e.g. Weihrauch [16] for the detail of the treatment of the computability of real numbers.

2.2 Algorithmic Information Theory

In the following we concisely review some definitions and results of algorithmic information theory [4,5]. A *computer* is a partial recursive function $C\colon \{0,1\}^* \to \{0,1\}^*$ such that $\mathrm{dom}\, C$ is a prefix-free set. For each computer C and each $s \in \{0,1\}^*$, $H_C(s)$ is defined by $H_C(s) = \min\{\, |p| \mid p \in \{0,1\}^* \,\&\, C(p) = s \,\}$ (may be ∞). A computer U is said to be *optimal* if for each computer C there exists $d \in \mathbb{N}$ with the following property; if $C(p)$ is defined, then there is a p' for which $U(p') = C(p)$ and $|p'| \leq |p| + d$. It is easy to see that there exists an optimal computer. Note that the class of optimal computers equals to the class of functions which are computed by *universal self-delimiting Turing machines* (see Chaitin [4] for the detail). We choose a particular optimal computer U as the standard one for use, and define $H(s)$ as $H_U(s)$, which is referred to as the *program-size complexity* of s or the *Kolmogorov complexity* of s. It follows that for every computer C there exists $d \in \mathbb{N}$ such that, every $s \in \{0,1\}^*$,

$$H(s) \leq H_C(s) + d. \tag{1}$$

Based on this we can show that there exists $c \in \mathbb{N}$ such that, for every $s \neq \lambda$,

$$H(s) \leq |s| + 2\log_2 |s| + c. \tag{2}$$

Chaitin's halting probability Ω is defined by $\Omega = \sum_{p \in \mathrm{dom}\, U} 2^{-|p|}$. For any $\alpha \in \mathbb{R}$, we say that α is *weakly Chaitin random* if there exists $c \in \mathbb{N}$ such that $n - c \leq H(\alpha_n)$ for all $n \in \mathbb{N}^+$ [4,5]. Then Chaitin [4] showed that Ω is

weakly Chaitin random. For any $\alpha \in \mathbb{R}$, we say that α is *Chaitin random* if $\lim_{n\to\infty} H(\alpha_n) - n = \infty$ [4,5]. It is then shown that, for every $\alpha \in \mathbb{R}$, α is weakly Chaitin random if and only if α is Chaitin random (see Chaitin [5] for the proof and historical detail). Thus Ω is Chaitin random.

2.3 Partial Randomness

In the works [12,13], we generalized the notion of the randomness of a real number so that *the degree of the randomness*, which is often referred to as *the partial randomness* recently [2,9,3], can be characterized by a real number T with $0 \leq T \leq 1$ as follows.

Definition 1 (weak Chaitin T-randomness). *Let $T \in \mathbb{R}$ with $T \geq 0$. For any $\alpha \in \mathbb{R}$, we say that α is weakly Chaitin T-random if there exists $c \in \mathbb{N}$ such that $Tn - c \leq H(\alpha_n)$ for all $n \in \mathbb{N}^+$.* □

Definition 2 (T-compressibility). *Let $T \in \mathbb{R}$ with $T \geq 0$. For any $\alpha \in \mathbb{R}$, we say that α is T-compressible if $H(\alpha_n) \leq Tn + o(n)$, which is equivalent to $\limsup_{n\to\infty} H(\alpha_n)/n \leq T$.* □

In the case of $T = 1$, the weak Chaitin T-randomness results in the weak Chaitin randomness. For every $T \in [0, 1]$ and every $\alpha \in \mathbb{R}$, if α is weakly Chaitin T-random and T-compressible, then

$$\lim_{n\to\infty} \frac{H(\alpha_n)}{n} = T. \tag{3}$$

The left-hand side of (3) is referred to as the *compression rate* of a real number α in general. Note, however, that (3) does not necessarily imply that α is weakly Chaitin T-random. Thus, the notion of partial randomness is a stronger representation of compression rate.

Definition 3 (Chaitin T-randomness, Tadaki [12,13]). *Let $T \in \mathbb{R}$ with $T \geq 0$. For any $\alpha \in \mathbb{R}$, we say that α is Chaitin T-random if $\lim_{n\to\infty} H(\alpha_n) - Tn = \infty$.* □

In the case of $T = 1$, the Chaitin T-randomness results in the Chaitin randomness. Obviously, for every $T \in [0, 1]$ and every $\alpha \in \mathbb{R}$, if α is Chaitin T-random, then α is weakly Chaitin T-random. However, in 2005 Reimann and Stephan [9] showed that, in the case of $T < 1$, the converse does not necessarily hold. This contrasts with the equivalence between the weak Chaitin randomness and the Chaitin randomness, each of which corresponds to the case of $T = 1$. Recently, Kjos-Hanssen [8] showed that the distinction between the weak Chaitin T-randomness and the Chaitin T-randomness has important applications to the research on the notion of T-capacitability and its related notions [7,10].

3 The Previous Results

In this section, we review some results of the statistical mechanical interpretation of AIT, developed by our former work [14]. We first introduce the notion of thermodynamic quantities into AIT in the following manner.

In statistical mechanics, the partition function $Z_{\rm sm}(T)$, free energy $F_{\rm sm}(T)$, energy $E_{\rm sm}(T)$, entropy $S_{\rm sm}(T)$, and specific heat $C_{\rm sm}(T)$ at temperature T are given as follows:

$$\begin{aligned} Z_{\rm sm}(T) &= \sum_{x\in X} e^{-\frac{E_x}{k_{\rm B}T}}, & F_{\rm sm}(T) &= -k_{\rm B}T\ln Z_{\rm sm}(T), \\ E_{\rm sm}(T) &= \frac{1}{Z_{\rm sm}(T)}\sum_{x\in X} E_x e^{-\frac{E_x}{k_{\rm B}T}}, & S_{\rm sm}(T) &= \frac{E_{\rm sm}(T)-F_{\rm sm}(T)}{T}, \\ & & C_{\rm sm}(T) &= \frac{d}{dT}E_{\rm sm}(T), \end{aligned} \tag{4}$$

where X is a complete set of energy eigenstates of a quantum system and E_x is the energy of an energy eigenstate x. The constant $k_{\rm B}$ is called the Boltzmann Constant, and the ln denotes the natural logarithm.[2]

We introduce the notion of thermodynamic quantities into AIT by performing Replacements 1 below for the thermodynamic quantities (4) in statistical mechanics.

Replacements 1

(i) Replace the complete set X of energy eigenstates x by the set $\mathrm{dom}\, U$ *of all programs p for U.*
(ii) Replace the energy E_x of an energy eigenstate x by the length $|p|$ of a program p.
(iii) Set the Boltzmann Constant $k_{\rm B}$ to $1/\ln 2$. □

For that purpose, we first choose a particular recursive enumeration $p_1, p_2, p_3, p_4, \ldots$ of the infinite r.e. set $\mathrm{dom}\, U$ as the standard one for use throughout the rest of this paper.[3] Then, motivated by the formulae (4) and taking into account Replacements 1, we introduce the notion of thermodynamic quantities into AIT as follows.

Definition 4 (thermodynamic quantities in AIT, [14]). *Let T be any real number with $T > 0$.*

[2] For the thermodynamic quantities in statistical mechanics, see e.g. Chapter 16 of [1] and Chapter 2 of [15]. To be precise, the partition function is not a thermodynamic quantity but a statistical mechanical quantity.

[3] Actually, the enumeration $\{p_i\}$ can be chosen quite arbitrarily, and the results of this paper is independent of the choice of $\{p_i\}$. For simplicity, however, we require $\{p_i\}$ to be a recursive enumeration of $\mathrm{dom}\, U$ in this paper.

(i) The partition function $Z(T)$ *at temperature* T *is defined as* $\lim_{k\to\infty} Z_k(T)$ *where*

$$Z_k(T) = \sum_{i=1}^{k} 2^{-\frac{|p_i|}{T}}.$$

(ii) The free energy $F(T)$ *at temperature* T *is defined as* $\lim_{k\to\infty} F_k(T)$ *where*

$$F_k(T) = -T \log_2 Z_k(T).$$

(iii) The energy $E(T)$ *at temperature* T *is defined as* $\lim_{k\to\infty} E_k(T)$ *where*

$$E_k(T) = \frac{1}{Z_k(T)} \sum_{i=1}^{k} |p_i|\, 2^{-\frac{|p_i|}{T}}.$$

(iv) The statistical mechanical entropy $S(T)$ *at temperature* T *is defined as* $\lim_{k\to\infty} S_k(T)$ *where*

$$S_k(T) = \frac{E_k(T) - F_k(T)}{T}.$$

(v) The specific heat $C(T)$ *at temperature* T *is defined as* $\lim_{k\to\infty} C_k(T)$ *where* $C_k(T) = E_k'(T)$*, the derived function of* $E_k(T)$*.* □

Note that $Z(1) = \Omega$ in particular. Then Theorems 1 and 2 below hold for these thermodynamic quantities in AIT.

Theorem 1 (properties of $Z(T)$ and $F(T)$, [12,13,14]). *Let* $T \in \mathbb{R}$*.*

(i) If $0 < T \leq 1$ *and* T *is computable, then each of* $Z(T)$ *and* $F(T)$ *converges and is weakly Chaitin* T*-random and* T*-compressible.*
(ii) If $1 < T$*, then* $Z(T)$ *and* $F(T)$ *diverge to* ∞ *and* $-\infty$*, respectively.* □

Theorem 2 (properties of $E(T)$, $S(T)$, and $C(T)$, [14]). *Let* $T \in \mathbb{R}$*.*

(i) If $0 < T < 1$ *and* T *is computable, then each of* $E(T)$*,* $S(T)$*, and* $C(T)$ *converges and is Chaitin* T*-random and* T*-compressible.*
(ii) If $1 \leq T$*, then both* $E(T)$ *and* $S(T)$ *diverge to* ∞*. In the case of* $T = 1$*,* $C(T)$ *diverges to* ∞*.*[4] □

The above two theorems show that if T is a computable real number with $T \in (0, 1)$ then the temperature T equals to the partial randomness (and therefore the compression rate) of the values of all the thermodynamic quantities in Definition 4.

Note that, in statistical mechanics or thermodynamics, among all thermodynamic quantities one of the most typical thermodynamic quantities is temperature itself. Thus, inspired by this fact in physics and the above observation on the role of the temperature T in our statistical mechanical interpretation of AIT, the following question arises naturally: Can the partial randomness of the temperature T equal to the temperature T itself in our statistical mechanical interpretation of AIT ? This question is rather self-referential. However, we can answer it affirmatively in the following form.

[4] It is still open whether $C(T)$ diverges or not in the case of $T > 1$.

Theorem 3 (fixed point theorem on partial randomness, [14]). *For every $T \in (0,1)$, if $Z(T)$ is computable, then T is weakly Chaitin T-random and T-compressible, and therefore $\lim_{n\to\infty} H(T_n)/n = T$.* □

Theorem 3 is just a fixed point theorem on partial randomness, where the computability of the value $Z(T)$ gives a sufficient condition for a real number $T \in (0,1)$ to be a fixed point on partial randomness. Thus, the above observation that the temperature T equals to the partial randomness of the values of the thermodynamic quantities in the statistical mechanical interpretation of AIT is further confirmed. In this paper, we confirm this observation much further by showing that fixed point theorems of the same form as Theorem 3 hold also for free energy $F(T)$, energy $E(T)$, and statistical mechanical entropy $S(T)$. For completeness, we include the proof of Theorem 3 in Appendix A.

4 The Main Results

The following three theorems are the main results of this paper.

Theorem 4 (fixed point theorem by free energy). *For every $T \in (0,1)$, if $F(T)$ is computable then T is weakly Chaitin T-random and T-compressible.* □

Theorem 5 (fixed point theorem by energy). *For every $T \in (0,1)$, if $E(T)$ is computable then T is Chaitin T-random and T-compressible.* □

Theorem 6 (fixed point theorem by statistical mechanical entropy). *For every $T \in (0,1)$, if $S(T)$ is computable then T is Chaitin T-random and T-compressible.* □

First, note that the weak Chaitin T-randomness of T in Theorems 3 is strengthen to the Chaitin T-randomness of T, in Theorems 5 and 6.

The proof of Theorem 4 uses Theorems 8, 10, and 11 below. On the other hand, the proofs of Theorems 5 and 6 use Theorems 9, 10, and 11 below. All these proofs also use Theorem 7 below, where the thermodynamic relations in statistical mechanics are recovered by the thermodynamic quantities of AIT. We describe the detail of the proofs of Theorems 4, 5, and 6 in the next section. Compared with the proof of Theorem 4, the proofs of Theorems 5 and 6 are more delicate.

Theorem 7 (thermodynamic relations)

(i) $F_k'(T) = -S_k(T)$, $E_k'(T) = C_k(T)$, *and* $S_k'(T) = C_k(T)/T$ *for every* $k \in \mathbb{N}^+$ *and every* $T \in (0,1)$.
(ii) $F'(T) = -S(T)$, $E'(T) = C(T)$, *and* $S'(T) = C(T)/T$ *for every* $T \in (0,1)$.
(iii) $S_k(T), C_k(T) \geq 0$ *for every* $k \in \mathbb{N}^+$ *and every* $T \in (0,1)$. *There exists* $k_0 \in \mathbb{N}^+$ *such that, for every* $k \geq k_0$ *and every* $T \in (0,1)$, $S_k(T), C_k(T) > 0$. *Moreover,* $S(T), C(T) > 0$ *for every* $T \in (0,1)$. □

The proof of Theorem 7 uses Lemma 1 below. For each $T \in (0,1)$, we define $W(T)$ and $Y(T)$ as $\lim_{k\to\infty} W_k(T)$ and $\lim_{k\to\infty} Y_k(T)$, respectively, where $W_k(T) = \sum_{i=1}^{k} |p_i|\, 2^{-\frac{|p_i|}{T}}$ and $Y_k(T) = \sum_{i=1}^{k} |p_i|^2\, 2^{-\frac{|p_i|}{T}}$.

Lemma 1

(i) For every $T \in (0,1)$, the limit values $Z(T)$, $W(T)$, and $Y(Z)$ exist, and are positive real numbers.

(ii) The sequence $\{Z_k(T)\}_k$ of functions of T is uniformly convergent on $(0,1)$ in the wider sense. The same holds for the sequences $\{W_k(T)\}_k$ and $\{Y_k(T)\}_k$.

(iii) The function $Z(T)$ of T is continuous on $(0,1)$. The same holds for the functions $W(T)$ and $Y(T)$.

Proof. (i) Suppose that T is an arbitrary real number with $T \in (0,1)$.

First, we show that $Y_k(T)$ converges as $k \to \infty$. Since $T < 1$, there is $l_0 \in \mathbb{N}^+$ such that

$$\frac{1}{T} - 2\frac{\log_2 l}{l} \geq 1$$

for all $l \geq l_0$. Then, since $\lim_{k\to\infty} |p_k| = \infty$, there is $k_0 \in \mathbb{N}^+$ such that $|p_i| \geq l_0$ for all $i > k_0$. Thus, we see that, for each $i > k_0$,

$$|p_i|^2\, 2^{-\frac{|p_i|}{T}} = 2^{-\left(\frac{1}{T} - 2\frac{\log_2 |p_i|}{|p_i|}\right)|p_i|} \leq 2^{-|p_i|}.$$

Hence, for each $k > k_0$,

$$Y_k(T) - Y_{k_0}(T) = \sum_{i=k_0+1}^{k} |p_i|^2\, 2^{-\frac{|p_i|}{T}} \leq \sum_{i=k_0+1}^{k} 2^{-|p_i|} < \Omega = Z(1).$$

Therefore, since $\{Y_k(T)\}_k$ is an increasing sequence of positive real numbers bounded to the above, it converges to a positive real number as $k \to \infty$, as desired.

Note that $0 < Z_k(T) \leq W_k(T) \leq Y_k(T)$ for every $k \in \mathbb{N}^+$, and the sequences $\{Z_k(T)\}_k$ and $\{W_k(T)\}_k$ of positive real numbers are increasing. It follows that $Z_k(T)$ and $W_k(T)$ converge to a positive real number as $k \to \infty$.

(ii) Note that, for every $k \in \mathbb{N}^+$ and every $t, T \in (0,1)$ with $t \leq T$,

$$0 < Z(t) - Z_k(t) = \sum_{i=k+1}^{\infty} 2^{-\frac{|p_i|}{t}} \leq \sum_{i=k+1}^{\infty} 2^{-\frac{|p_i|}{T}} = Z(T) - Z_k(T).$$

It follows that the sequence $\{Z_k(T)\}_k$ of functions of T is uniformly convergent on $(0,1)$ in the wider sense. In the same manner, we can show that the sequences $\{W_k(T)\}_k$ and $\{Y_k(T)\}_k$ are uniformly convergent on $(0,1)$ in the wider sense.

(iii) Note that, for each $k \in \mathbb{N}^+$, the mapping $(0,1) \ni T \mapsto Z_k(T)$ is a continuous function. It follows from Lemma 1 (ii) that the function $Z(T)$ of T is continuous on $(0,1)$. In the same manner, we can show that the functions $W(T)$ and $Y(T)$ of T are continuous on $(0,1)$. □

Proof (of Theorem 7). (i) First, from Definition 4 we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$F_k(T) = -T \log_2 Z_k(T),$$

$$E_k(T) = \frac{W_k(T)}{Z_k(T)}, \tag{5}$$

$$S_k(T) = \frac{W_k(T)}{T Z_k(T)} + \log_2 Z_k(T), \tag{6}$$

$$Z_k'(T) = \frac{\ln 2}{T^2} W_k(T), \tag{7}$$

$$W_k'(T) = \frac{\ln 2}{T^2} Y_k(T). \tag{8}$$

Thus, by straightforward differentiation, we can check that the relations of Theorem 7 (i) hold. For example, it follows from (6) and (5) that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$S_k'(T) = \frac{1}{T} E_k'(T) - \frac{1}{T^2} \frac{W_k(T)}{Z_k(T)} + \frac{1}{\ln 2} \frac{Z_k'(T)}{Z_k(T)}.$$

Using the definition $C_k(T) = E_k'(T)$ and the equation (7) we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$, $S_k'(T) = C_k(T)/T$.

(ii) From (5), (7), (8), and the definition $C_k(T) = E_k'(T)$, we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$C_k(T) = \frac{\ln 2}{T^2} \left\{ \frac{Y_k(T)}{Z_k(T)} - \left(\frac{W_k(T)}{Z_k(T)} \right)^2 \right\}. \tag{9}$$

Using Lemma 1 above and the equations (6) and (9), we can check that the sequences $\{-S_k(T)\}_k$, $\{C_k(T)\}_k$ and $\{C_k(T)/T\}_k$ of functions of T are uniformly convergent on $(0,1)$ in the wider sense. Thus, Theorem 7 (ii) follows immediately from Theorem 7 (i).

(iii) From (6) we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$S_k(T) = -\sum_{i=1}^{k} \frac{2^{-\frac{|p_i|}{T}}}{Z_k(T)} \log_2 \frac{2^{-\frac{|p_i|}{T}}}{Z_k(T)}.$$

Thus, $S_k(T) \geq 0$ for every $k \in \mathbb{N}^+$. We also see that, for every $k \geq 2$ and every $T \in (0,1)$,

$$S_k(T) \geq -\frac{2^{-\frac{|p_1|}{T}}}{Z_k(T)} \log_2 \frac{2^{-\frac{|p_1|}{T}}}{Z_k(T)} > 0.$$

Hence, for every $T \in (0,1)$,

$$S(T) \geq -\frac{2^{-\frac{|p_1|}{T}}}{Z(T)} \log_2 \frac{2^{-\frac{|p_1|}{T}}}{Z(T)} > 0.$$

On the other hand, from (9) we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$C_k(T) = \frac{\ln 2}{T^2} \sum_{i=1}^{k} \{|p_i| - E_k(T)\}^2 \frac{2^{-\frac{|p_i|}{T}}}{Z_k(T)}. \tag{10}$$

Thus, $C_k(T) \geq 0$ for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$. We note that there exists $l \in \mathbb{N}^+$ such that $|p_l| \leq |p_i|$ for every $i \in \mathbb{N}^+$. It is then easy to see that there exists $k_0 \in \mathbb{N}^+$ such that, for every $k \geq k_0$ and every $T \in (0,1)$, $|p_l| < E_k(T)$. This is because there exists $i \in \mathbb{N}^+$ such that $|p_l| < |p_i|$. Thus, by (10) we see that, for every $k \geq \max\{l, k_0\}$ and every $T \in (0,1)$,

$$C_k(T) \geq \frac{\ln 2}{T^2} \{|p_l| - E_k(T)\}^2 \frac{2^{-\frac{|p_l|}{T}}}{Z_k(T)} > 0. \tag{11}$$

It is also easy to see that $|p_l| < E(T)$ for every $T \in (0,1)$. It follows from (11) that $C(T) > 0$ for every $T \in (0,1)$. □

Theorem 8. *Let $f\colon (0,1) \to \mathbb{R}$. Suppose that f is increasing and there exists $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ which satisfies the following four conditions:*

(i) $\lim_{k\to\infty} g(T,k) = f(T)$ *for every* $T \in (0,1)$.
(ii) $\{(q,r,k) \in \mathbb{Q} \times (\mathbb{Q} \cap (0,1)) \times \mathbb{N}^+ \mid q < g(r,k)\}$ *is an r.e. set.*
(iii) For every $T \in (0,1)$, *there exist* $a \in \mathbb{N}$, $k_0 \in \mathbb{N}^+$, *and* $t \in (T,1)$ *such that, for every* $k \geq k_0$ *and every* $x \in (T,t)$, $g(x,k) - g(T,k) \leq 2^a(x-T)$.
(iv) For every $T \in (0,1)$, *there exist* $b \in \mathbb{N}$ *and* $k_1 \in \mathbb{N}^+$ *such that, for every* $k \geq k_1$,

$$2^{-\frac{|p_{k+1}|}{T} - b} \leq g(T,k+1) - g(T,k).$$

Then, for every $T \in (0,1)$, *if* $f(T)$ *is right-computable then* T *is weakly Chaitin* T*-random.*

Proof. The proof of Theorem 8 is obtained by slightly simplifying the proof of Theorem 9 below. □

Theorem 9. *Let $f\colon (0,1) \to \mathbb{R}$. Suppose that f is increasing and there exists $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ which satisfies the following four conditions:*

(i) $\lim_{k\to\infty} g(T,k) = f(T)$ *for every* $T \in (0,1)$.
(ii) $\{(q,r,k) \in \mathbb{Q} \times (\mathbb{Q} \cap (0,1)) \times \mathbb{N}^+ \mid q < g(r,k)\}$ *is an r.e. set.*
(iii) For every $T \in (0,1)$, *there exist* $a \in \mathbb{N}$, $k_0 \in \mathbb{N}^+$, *and* $t \in (T,1)$ *such that, for every* $k \geq k_0$ *and every* $x \in (T,t)$, $g(x,k) - g(T,k) \leq 2^a(x-T)$.
(iv) For every $T \in (0,1)$, *there exist* $b \in \mathbb{N}$, $c \in \mathbb{N}^+$, *and* $k_1 \in \mathbb{N}^+$ *such that, for every* $k \geq k_1$,

$$|p_{k+1}|^c 2^{-\frac{|p_{k+1}|}{T} - b} \leq g(T,k+1) - g(T,k).$$

Then, for every $T \in (0,1)$, *if* $f(T)$ *is right-computable then* T *is Chaitin* T*-random.*

Proof. Suppose that $f(T)$ is right-computable and $T \in (0,1)$. Then there exists a total recursive function $h\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $f(T) \leq h(m)$ for all $m \in \mathbb{N}^+$ and $\lim_{m\to\infty} h(m) = f(T)$.

Since the condition (iii) holds for g, there exist $a \in \mathbb{N}$, $k_0 \in \mathbb{N}^+$, and $t \in (T,1)$ such that

$$g(x,k) - g(T,k) \leq 2^a(x-T) \tag{12}$$

for every $k \geq k_0$ and every $x \in (T,t)$. We choose any one $n_0 \in \mathbb{N}^+$ such that $0.T_n{+}2^{-n} < t$ for all $n \geq n_0$. Such n_0 exists since $T < t$ and $\lim_{n\to\infty} 0.T_n{+}2^{-n} = T$. Since T_n is the first n bits of the base-two expansion of T with infinitely many zeros, we further see that $T < 0.T_n + 2^{-n} < t$ for all $n \geq n_0$.

On the other hand, since the condition (iv) holds for g, there exist $b \in \mathbb{N}$, $c \in \mathbb{N}^+$, and $k_1 \in \mathbb{N}^+$ such that

$$|p_{k+1}|^c 2^{-\frac{|p_{k+1}|}{T}-b} \leq g(T,k+1) - g(T,k)$$

for every $k \geq k_1$. Without loss of generality, we can assume that $k_1 = k_0$. Thus, since $g(T,k)$ is increasing on k with $k \geq k_0$ and the condition (i) holds,

$$|p_i|^c\, 2^{-\frac{|p_i|}{T}-b} < f(T) - g(T,k) \tag{13}$$

if $i > k \geq k_0$.

Now, given T_n with $n \geq n_0$, one can effectively find $k_e, m_e \in \mathbb{N}^+$ such that $k_e \geq k_0$ and $h(m_e) < g(0.T_n{+}2^{-n}, k_e)$. This is possible because $f(T) < f(0.T_n{+}2^{-n})$, $\lim_{k\to\infty} g(0.T_n + 2^{-n}, k) = f(0.T_n + 2^{-n})$, and the condition (ii) holds for g. It follows from $f(T) \leq h(m_e)$ and (12) that $f(T) - g(T,k_e) < g(0.T_n + 2^{-n}, k_e) - g(T,k_e) \leq 2^{a-n}$. It follows from (13) that, for every $i > k_e$, $|p_i|^c\, 2^{-\frac{|p_i|}{T}-b} < 2^{a-n}$ and therefore $cT\log_2|p_i| - (a+b)T < |p_i| - Tn$. Thus, by calculating the set $\{\, U(p_i) \mid i \leq k_e \,\}$ and picking any one finite binary string s which is not in this set, one can then obtain $s \in \{0,1\}^*$ such that $cT\log_2 H(s) - (a{+}b)T < H(s) - Tn$.

Hence, there exists a partial recursive function $\Psi\colon \{0,1\}^* \to \{0,1\}^*$ such that

$$cT\log_2 H(\Psi(T_n)) - (a+b)T < H(\Psi(T_n)) - Tn$$

for all $n \geq n_0$. Applying this inequality to itself, we have $cT\log_2 n < H(\Psi(T_n)) - Tn + O(1)$, for all $n \in \mathbb{N}^+$. On the other hand, using (1), there is $c_\Psi \in \mathbb{N}$ such that $H(\Psi(T_n)) \leq H(T_n) + c_\Psi$ for all $n \geq n_0$. It follows that $cT\log_2 n < H(T_n) - Tn + O(1)$. Hence, T is Chaitin T-random. □

Theorem 10. *Let $f\colon (0,1) \to \mathbb{R}$. Suppose that f is increasing and there exists $g\colon (0,1)\times\mathbb{N}^+ \to \mathbb{R}$ which satisfies the following three conditions:*

(i) For every $T \in (0,1)$, $\lim_{k\to\infty} g(T,k) = f(T)$.
(ii) For every $T_1, T_2 \in (0,1)$ with $T_1 < T_2$, there exists $k_0 \in \mathbb{N}^+$ such that, for every $k \geq k_0$ and every $x \in [T_1, T_2]$, $g(x,k) \leq f(x)$.
(iii) $\{(q,r,k) \in \mathbb{Q}\times(\mathbb{Q}\cap(0,1))\times\mathbb{N}^+ \mid q < g(r,k)\}$ is an r.e. set.

Then, for every $T \in (0,1)$, if $f(T)$ is right-computable then T is also right-computable.

Proof. Suppose that $T \in (0,1)$. We choose any $t_1, t_2 \in \mathbb{Q}$ with $0 < t_1 < T < t_2 < 1$. Then, since the condition (ii) holds for g, there exists $k_0 \in \mathbb{N}^+$ such that $g(x,k) \leq f(x)$ for every $k \geq k_0$ and every $x \in [t_1, t_2]$. Suppose further that $f(T)$ is right-computable. Then there exists a total recursive function $h\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $f(T) \leq h(m)$ for all $m \in \mathbb{N}^+$ and $\lim_{m\to\infty} h(m) = f(T)$. Thus, since f is increasing and the condition (i) holds for g, we see that, for every $r \in \mathbb{Q} \cap [t_1, t_2]$, $T < r$ if and only if $\exists\, m\, \exists\, k \geq k_0\ h(m) < g(r,k)$. Since the condition (iii) holds for g, the set $\{\, r \in \mathbb{Q} \cap [t_1, t_2] \mid \exists\, m\, \exists\, k \geq k_0\ h(m) < g(r,k) \,\}$ is r.e. and therefore the set $\{\, r \in \mathbb{Q} \cap [t_1, t_2] \mid T < r \,\}$ is r.e. It follows from $T \in (t_1, t_2)$ that T is right-computable. □

Theorem 11. *Let $f\colon (0,1) \to \mathbb{R}$. Suppose that there exists $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ which satisfies the following six conditions:*

(i) *For every $T \in (0,1)$, $\lim_{k\to\infty} g(T,k) = f(T)$.*
(ii) *For every $T \in (0,1)$, there exists $k_0 \in \mathbb{N}^+$ such that, for every $k \geq k_0$, $g(T,k) < f(T)$.*
(iii) *For every $T \in (0,1)$, there exist $a \in \mathbb{N}$, $k_1 \in \mathbb{N}^+$, and $t \in (T,1)$ such that, for every $k \geq k_1$ and every $x \in (T,t)$, $g(x,k) - g(T,k) \geq 2^{-a}(x-T)$.*
(iv) *For every $T \in (0,1)$, there exist $b \in \mathbb{N}$, $c \in \mathbb{N}$, and $k_2 \in \mathbb{N}^+$ such that, for every $k \geq k_2$,*

$$g(T,k+1) - g(T,k) \leq |p_{k+1}|^b\, 2^{-|p_{k+1}|/T+c}.$$

(v) *For each $k \in \mathbb{N}^+$, the mapping $(0,1) \ni T \mapsto g(T,k)$ is a continuous function.*
(vi) *$\{(q,r,k) \in \mathbb{Q} \times (\mathbb{Q} \cap (0,1)) \times \mathbb{N}^+ \mid q > g(r,k)\}$ is an r.e. set.*

Then, for every $T \in (0,1)$, if $f(T)$ is left-computable and T is right-computable, then T is T-compressible.

Proof. Suppose that $T \in (0,1)$. Since the condition (ii) holds for g, there exists $k_0 \in \mathbb{N}^+$ such that

$$g(T,k) < f(T) \tag{14}$$

for every $k \geq k_0$. Since the condition (iii) holds for g, there exist $a \in \mathbb{N}$, $k_1 \in \mathbb{N}^+$, and $t \in (T,1)$ such that

$$g(x,k) - g(T,k) \geq 2^{-a}(x-T) \tag{15}$$

for every $k \geq k_1$ and every $x \in (T,t)$. Since the condition (iv) holds for g, there exist $b \in \mathbb{N}$, $c \in \mathbb{N}$, and $k_2 \in \mathbb{N}^+$ such that

$$g(T,k+1) - g(T,k) \leq |p_{k+1}|^b\, 2^{-|p_{k+1}|/T+c} \tag{16}$$

for every $k \geq k_2$. Without loss of generality, we can assume that $k_0 = k_1 = k_2$.

Suppose further that T is right-computable and $f(T)$ is left-computable. Then there exists a total recursive function $A\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $T < A(l) < t$ for

all $l \in \mathbb{N}^+$ and $\lim_{l\to\infty} A(l) = T$, and there exists a total recursive function $B\colon \mathbb{N}^+ \to \mathbb{Q}$ such that $B(m) \leq f(T)$ for all $m \in \mathbb{N}^+$ and $\lim_{m\to\infty} B(m) = f(T)$.

Let u be an arbitrary computable real number with $T < u < 1$, and let $\beta = \sum_{i=1}^{\infty} |p_i|^b\, 2^{-|p_i|/u}$. Note that this limit exists and is weakly Chaitin u-random (see Theorem 3.2 (a) of [13] and Theorem 3 (i) of [14]). Thus, the base-two expansion of β contains infinitely many zeros and infinitely many ones.

Given n and $\beta_{\lceil Tn/u\rceil}$ (i.e., the first $\lceil Tn/u\rceil$ bits of the base-two expansion of $\beta - \lfloor\beta\rfloor$), one can effectively find $k_e \in \mathbb{N}^+$ such that $k_e \geq k_0$ and

$$0.\beta_{\lceil Tn/u\rceil} + \lfloor\beta\rfloor < \sum_{i=1}^{k_e} |p_i|^b\, 2^{-\frac{|p_i|}{u}}.$$

This is possible since $0.\beta_{\lceil Tn/u\rceil} + \lfloor\beta\rfloor < \beta$ and $\beta = \lim_{k\to\infty} \sum_{i=1}^{k} |p_i|^b\, 2^{-|p_i|/u}$. Since $\beta - (0.\beta_{\lceil Tn/u\rceil} + \lfloor\beta\rfloor) \leq 2^{-\lceil Tn/u\rceil} \leq 2^{-Tn/u}$, it is then shown that

$$\sum_{i=k_e+1}^{\infty} |p_i|^b\, 2^{-\frac{|p_i|}{u}} = \beta - \sum_{i=1}^{k_e} |p_i|^b\, 2^{-\frac{|p_i|}{u}} < 2^{-Tn/u}.$$

Raising both ends of this inequality to the power u/T and using the inequality $x^z + y^z \leq (x+y)^z$ for real numbers $x, y > 0$ and $z \geq 1$, we have

$$\sum_{i=k_e+1}^{\infty} |p_i|^b\, 2^{-\frac{|p_i|}{T}} \leq \sum_{i=k_e+1}^{\infty} |p_i|^{\frac{bu}{T}}\, 2^{-\frac{|p_i|}{T}} < 2^{-n}.$$

Using (16) and the condition (i), it follows that

$$f(T) - g(T,k_e) < \sum_{i=k_e+1}^{\infty} |p_i|^b\, 2^{-\frac{|p_i|}{T}+c} < 2^{c-n}. \tag{17}$$

On the other hand, since the condition (v) holds for g, $g(T,k_e)=\lim_{l\to\infty} g(A(l),k_e)$. Obviously, $g(T,k_e) < f(T)$ by (14). Thus, since the condition (vi) holds for g, one can then effectively find $l_e, m_e \in \mathbb{N}^+$ such that $g(A(l_e),k_e) < B(m_e)$. It follows from (17) and (15) that

$$2^{c-n} > f(T) - g(T,k_e) \geq B(m_e) - g(T,k_e) > g(A(l_e),k_e) - g(T,k_e) \geq 2^{-a}(A(l_e) - T).$$

Thus, $0 < A(l_e) - T < 2^{a+c-n}$. Let r_n be the first n bits of the base-two expansion of the rational number $A(l_e)$ with infinitely many zeros. Then $|\, A(l_e) - 0.r_n\,| < 2^{-n}$. It follows from $|\, T - 0.T_n\,| < 2^{-n}$ that $|\, 0.T_n - 0.r_n\,| < (2^{a+c}+2)2^{-n}$. Hence, $T_n = r_n$, $r_n \pm 1$, $r_n \pm 2$, $\ldots$, $r_n \pm (2^{a+c}+1)$, where T_n and r_n are regarded as a dyadic integer. Thus, there are still $2^{a+c+1}+3$ possibilities of T_n, so that one needs only $a+c+3$ bits more in order to determine T_n.

Thus, there exists a partial recursive function $\Phi\colon \mathbb{N}^+ \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that, for every $n \in \mathbb{N}^+$, there exists $s \in \{0,1\}^*$ with the properties that $|s| = a+c+3$ and $\Phi(n, \beta_{\lceil Tn/u\rceil}, s) = T_n$. It follows from (2) that $H(T_n) \leq |\beta_{\lceil Tn/u\rceil}| + o(n) \leq Tn/u + o(n)$, which implies that T is T/u-compressible. Since u is an arbitrary computable real number with $T < u < 1$, it follows that T is T-compressible. □

5 The Proofs of the Main Results

In this section we complete the proofs of our main results; Theorems 4, 5, and 6.

5.1 The Proof of Theorem 4

We first complete the proof of Theorem 4, based on Theorems 7, 8, 10, and 11, as follows.

Let $f\colon (0,1) \to \mathbb{R}$ with $f(T) = -F(T)$, and let $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ with $g(T,k) = -F_k(T)$. First, it follows from Theorem 7 (ii) and (iii) that f is increasing.

Obviously, $\lim_{k\to\infty} g(T,k) = f(T)$ for every $T \in (0,1)$. Using the mean value theorem we see that, for every $T \in (0,1)$ and every $k \in \mathbb{N}^+$,

$$\frac{2^{-\frac{|p_{k+1}|}{T}}}{Z_{k+1}(T)} < \ln Z_{k+1}(T) - \ln Z_k(T) < \frac{2^{-\frac{|p_{k+1}|}{T}}}{Z_k(T)}. \tag{18}$$

It follows that, for every $T \in (0,1)$ and every $k \in \mathbb{N}^+$, $g(T,k) < g(T,k+1)$ and therefore $g(T,k) < f(T)$. At this point, the conditions (i) and (ii) of Theorem 8, all conditions of Theorem 10, and the conditions (i), (ii), (v), and (vi) of Theorem 11 hold for g.

Using (18) we see that, for every $T \in (0,1)$ and every $k \in \mathbb{N}^+$,

$$\frac{T2^{-\frac{|p_{k+1}|}{T}}}{Z_{k+1}(T)\ln 2} < g(T,k+1) - g(T,k) < \frac{T2^{-\frac{|p_{k+1}|}{T}}}{Z_k(T)\ln 2}.$$

Thus, the condition (iv) of Theorem 8 and the condition (iv) of Theorem 11 hold for g.

Using the mean value theorem and Theorem 7 (i) and (iii), we see that

$$S_k(T)(x-T) \le g(x,k) - g(T,k) \le S_k(t)(x-T) \tag{19}$$

for every $k \in \mathbb{N}^+$ and every $T, x, t \in (0,1)$ with $T < x < t$. On the other hand, we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$E_{k+1}(T) - E_k(T) = \frac{Z_k(T)\,|p_{k+1}| - W_k(T)}{Z_{k+1}(T)Z_k(T)} 2^{-\frac{|p_{k+1}|}{T}}.$$

Recall here that, for every $T \in (0,1)$, $\lim_{k\to\infty} Z_k(T)$ and $\lim_{k\to\infty} W_k(T)$ exist and are positive by Lemma 1 (i). It follows from $\lim_{k\to\infty} |p_{k+1}| = \infty$ that, for every $T \in (0,1)$, there exists $k_0 \in \mathbb{N}^+$ such that, for every $k \ge k_0$, $E_k(T) < E_{k+1}(T)$ and therefore $S_k(T) < S_{k+1}(T)$ by (18). Using Theorem 7 (iii), we see that, for every $T \in (0,1)$, there exists $k_1 \in \mathbb{N}^+$ such that, for every $k \ge k_1$, $0 < S_{k_1}(T) \le S_k(T) < S(T)$. Thus, using (19), for every $T, t \in (0,1)$ with $T < t$, there exists $k_2 \in \mathbb{N}^+$ such that $S_{k_2}(T) > 0$ and for every $k \ge k_2$ and every $x \in (T,t)$, $S_{k_2}(T)(x-T) \le g(x,k) - g(T,k) < S(t)(x-T)$. Therefore, the condition (iii) of Theorem 8 and the condition (iii) of Theorem 11 hold for g.

Thus, Theorem 8, Theorem 10, and Theorem 11 result in the following three theorems, respectively.

Theorem 12. *For every $T \in (0,1)$, if $F(T)$ is left-computable then T is weakly Chaitin T-random.* □

Theorem 13. *For every $T \in (0,1)$, if $F(T)$ is left-computable then T is right-computable.* □

Theorem 14. *For every $T \in (0,1)$, if both $F(T)$ and T are right-computable then T is T-compressible.* □

Theorem 4 follows immediately from these three theorems.

5.2 The Proof of Theorem 5

We complete the proof of Theorem 5, based on Theorems 7, 9, 10, and 11, as follows.

Let $f\colon (0,1) \to \mathbb{R}$ with $f(T) = E(T)$, and let $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ with $g(T,k) = E_k(T)$. First, by Theorem 7 (ii) and (iii), we see that $E'(T) = C(T) > 0$ for every $T \in (0,1)$. Thus f is increasing.

Obviously, $\lim_{k\to\infty} g(T,k) = f(T)$ for every $T \in (0,1)$. At this point, the conditions (i) and (ii) of Theorem 9, the conditions (i) and (iii) of Theorem 10, and the conditions (i), (v), and (vi) of Theorem 11 hold for g.

We see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$,

$$E_{k+1}(T) - E_k(T) = \frac{Z_k(T)\,|p_{k+1}| - W_k(T)}{Z_{k+1}(T)Z_k(T)} 2^{-\frac{|p_{k+1}|}{T}}. \tag{20}$$

Recall here that, for every $T \in (0,1)$, $\lim_{k\to\infty} Z_k(T)$ and $\lim_{k\to\infty} W_k(T)$ exist and are positive by Lemma 1 (i). It follows from $\lim_{k\to\infty} |p_{k+1}| = \infty$ that, for every $T \in (0,1)$, there exist $a \in \mathbb{N}$, $b \in \mathbb{N}$, and $k_0 \in \mathbb{N}^+$ such that, every $k \geq k_0$,

$$|p_{k+1}|\, 2^{-\frac{|p_{k+1}|}{T} - a} \leq g(T,k+1) - g(T,k) \leq |p_{k+1}|\, 2^{-\frac{|p_{k+1}|}{T} + b}. \tag{21}$$

Thus, the condition (iv) of Theorem 9 and the condition (iv) of Theorem 11 hold for g. It follows from (21) that, for every $T \in (0,1)$, there exists $k_0 \in \mathbb{N}^+$ such that, every $k \geq k_0$, $g(T,k) < g(T,k+1)$ and therefore $g(T,k) < f(T)$. Thus, the condition (ii) of Theorem 11 holds for g.

Using Lemma 1 (ii) and (iii) in addition to Lemma 1 (i), we can show a stronger statement than the inequalities (21). The stronger statement for the lower bound of (21) is needed here. That is, based on (20), Lemma 1, and $\lim_{k\to\infty} |p_{k+1}| = \infty$, we can show that, for every $T_1, T_2 \in (0,1)$ with $T_1 < T_2$, there exist $a \in \mathbb{N}$ and $k_0 \in \mathbb{N}^+$ such that, every $k \geq k_0$ and every $x \in [T_1, T_2]$,

$$|p_{k+1}|\, 2^{-\frac{|p_{k+1}|}{x} - a} \leq g(x,k+1) - g(x,k).$$

It follows that the condition (ii) of Theorem 10 holds for g.

Now, using the mean value theorem and Theorem 7 (i), we see that, for every $k \in \mathbb{N}^+$ and every $T, x, t \in (0,1)$ with $T < x < t$, there exists $y \in (T,x)$ such that

$$g(x,k) - g(T,k) = C_k(y)(x - T). \tag{22}$$

On the other hand, using (9) we see that, for every $k \in \mathbb{N}^+$ and every $T \in (0,1)$, $C_{k+1}(T) - C_k(T)$ is calculated as

$$\frac{\ln 2}{T^2} \frac{2^{-\frac{|p_{k+1}|}{T}}}{Z_{k+1}(T)} \left[|p_{k+1}|^2 - \left\{ \frac{W_{k+1}(T)}{Z_{k+1}(T)} + \frac{W_k(T)}{Z_k(T)} \right\} |p_{k+1}| + \left\{ \frac{W_{k+1}(T)}{Z_{k+1}(T)} + \frac{W_k(T)}{Z_k(T)} \right\} \frac{W_k(T)}{Z_k(T)} - \frac{Y_k(T)}{Z_k(T)} \right].$$

Thus, based on Lemma 1 and $\lim_{k\to\infty} |p_{k+1}| = \infty$, we can show that, for every $T_1, T_2 \in (0,1)$ with $T_1 < T_2$, there exist $a \in \mathbb{N}$ and $k_0 \in \mathbb{N}^+$ such that, every $k \geq k_0$ and every $y \in [T_1, T_2]$,

$$|p_{k+1}|^2 \, 2^{-\frac{|p_{k+1}|}{y} - a} \leq C_{k+1}(y) - C_k(y).$$

It follows from Theorem 7 (iii) that, for every $T_1, T_2 \in (0,1)$ with $T_1 < T_2$, there exist $a \in \mathbb{N}$ and $k_0 \in \mathbb{N}^+$ such that, every $k \geq k_0$ and every $y \in [T_1, T_2]$,

$$0 < \min C_{k_0}([T_1, T_2]) \leq C_k(y) < \max C([T_1, T_2]), \tag{23}$$

where $\min C_{k_0}([T_1, T_2]) = \min\{ C_{k_0}(z) \mid z \in [T_1, T_2] \}$ and $\max C([T_1, T_2]) = \max\{ C(z) \mid z \in [T_1, T_2] \}$. In particular, $\max C([T_1, T_2])$ exists. This is because the function $C(T)$ of T is continuous on $(0,1)$ by Lemma 1 and (9). It follows from (22) and (23) that, for every $T, t \in (0,1)$ with $T < t$, there exist $a \in \mathbb{N}$, $b \in \mathbb{N}$, and $k_0 \in \mathbb{N}^+$ such that, for every $k \geq k_0$ and every $x \in (T,t)$, $2^{-a}(x-T) \leq g(x,k) - g(T,k) < 2^b(x-T)$. Therefore, the condition (iii) of Theorem 9 and the condition (iii) of Theorem 11 hold for g.

Thus, Theorem 9, Theorem 10, and Theorem 11 result in the following three theorems, respectively.

Theorem 15. *For every $T \in (0,1)$, if $E(T)$ is right-computable then T is Chaitin T-random.* □

Theorem 16. *For every $T \in (0,1)$, if $E(T)$ is right-computable then T is also right-computable.* □

Theorem 17. *For every $T \in (0,1)$, if $E(T)$ is left-computable and T is right-computable, then T is T-compressible.* □

Theorem 5 follows immediately from these three theorems.

5.3 The Proof of Theorem 6

In a similar manner to the proof of Theorem 5 described in the previous subsection, we can prove Theorem 6, based on Theorems 7, 9, 10, and 11. It is easy to convert the proof of Theorem 5 into the the proof of Theorem 6, because of the similarity between $E'_k(T) = C_k(T)$ and $S'_k(T) = C_k(T)/T$ given in Theorem 7 (i).

6 Some Properties of the Sufficient Conditions

In this section, we investigate some properties of the sufficient conditions for T to be a fixed point in the fixed point theorems on partial randomness.

First note that the function $F(T)$ of T is a decreasing continuous function on $(0,1)$ by Theorem 7 (ii) and (iii), and the set of all computable real numbers is dense in $(0,1)$. Thus, we can easily show the following theorem for the sufficient condition of Theorem 4. The exactly same theorem holds for each of $Z(T)$, $E(T)$, and $S(T)$ also.

Theorem 18. *The set* $\{\, T \in (0,1) \mid F(T) \text{ is computable} \,\}$ *is dense in* $(0,1)$. □

Since the relation $F(T) = -T \log_2 Z(T)$ holds from Definition 4, we can show the following theorem.

Theorem 19. *There does not exist* $T \in (0,1)$ *such that both* $Z(T)$ *and* $F(T)$ *are computable.*

Proof. Contrarily, assume that both $Z(T)$ and $F(T)$ are computable for some $T \in (0,1)$. Since $F(T) = -T \log_2 Z(T)$ and $0 < Z(T) < 1$, it is easy to see that T is computable. It follows from Theorem 1 (i) that $Z(T)$ is weakly Chaitin T-random. However, this contradicts the assumption that $Z(T)$ is computable, and the result follows. □

Thus, the computability of $F(T)$ gives completely different fixed points from the computability of $Z(T)$. This implies that neither the computability of $Z(T)$ nor the computability of $F(T)$ is the necessary condition for T to be a fixed point on partial randomness at all.

In a similar manner, we can prove the following two theorems using the relations $S(T) = E(T)/T + \log_2 Z(T)$ and $S(T) = (E(T) - F(T))/T$, respectively.

Theorem 20. *There does not exist* $T \in (0,1)$ *such that* $Z(T)$*,* $E(T)$*, and* $S(T)$ *are all computable.* □

Theorem 21. *There does not exist* $T \in (0,1)$ *such that* $F(T)$*,* $E(T)$*, and* $S(T)$ *are all computable.* □

Using the property of a fixed point in the fixed point theorems, we can show the following theorem.

Theorem 22. $S_a \cap S_b = \emptyset$ *for any distinct computable real numbers* $a, b \in (0,1]$*, where* $S_a = \{\, T \in (0,1) \mid Z(aT) \text{ is computable} \,\}$.

Proof. Let $T \in (0,1)$, and let a be a computable real number with $a \in (0,1]$. Suppose that $Z(aT)$ is computable. Then, by Theorem 3, $\lim_{n\to\infty} H((aT)_n)/n = aT$. It follows from $H((aT)_n) = H(T_n) + O(1)$ that $\lim_{n\to\infty} H(T_n)/n = aT$.

Thus, for every computable real numbers $a, b \in (0,1]$, if $S_a \cap S_b \neq \emptyset$ then $a = b$. □

As a corollary of Theorem 22, we have the following, for example.

Corollary 1. *For every $T \in (0,1)$, if $Z(T)$ is computable, then $Z(T/n)$ is not computable for every $n \in \mathbb{N}$ with $n \geq 2$. In other words, for every $T \in (0,1)$, if the sum $\sum_{i=1}^{\infty} 2^{-|p_i|/T}$ is computable, then the corresponding power sum $\sum_{i=1}^{\infty} \left(2^{-|p_i|/T}\right)^n$ is not computable for every $n \in \mathbb{N}$ with $n \geq 2$.* □

Acknowledgments. This work was supported by KAKENHI, Grant-in-Aid for Scientific Research (C) (20540134), SCOPE of the Ministry of Internal Affairs and Communications of Japan, and CREST of the Japan Science and Technology Agency.

References

1. Callen, H.B.: Thermodynamics and an Introduction to Thermostatistics, 2nd edn. John Wiley & Sons, Inc., Singapore (1985)
2. Calude, C.S., Staiger, L., Terwijn, S.A.: On partial randomness. Ann. Pure Appl. Logic 138, 20–30 (2006)
3. Calude, C.S., Stay, M.A.: Natural halting probabilities, partial randomness, and zeta functions. Inform. and Comput. 204, 1718–1739 (2006)
4. Chaitin, G.J.: A theory of program size formally identical to information theory. J. Assoc. Comput. Mach. 22, 329–340 (1975)
5. Chaitin, G.J.: Algorithmic Information Theory. Cambridge University Press, Cambridge (1987)
6. Downey, R.G., Hirschfeldt, D.R.: Algorithmic randomness and complexity. Springer, Heidelberg (to appear)
7. Kjos-Hanssen, B.: Infinite subsets of random sets of integers. Math. Res. Lett. (to appear)
8. Kjos-Hanssen, B.: Private communication (September 2008)
9. Reimann, J., Stephan, F.: On hierarchies of randomness tests. In: Proceedings of the 9th Asian Logic Conference. World Scientific Publishing, Novosibirsk (2005)
10. Reimann, J.: Effectively closed sets of measures and randomness. Ann. Pure Appl. Logic (to appear)
11. Ruelle, D.: Statistical Mechanics, Rigorous Results, 3rd edn. Imperial College Press and World Scientific Publishing Co. Pte. Ltd., Singapore (1999)
12. Tadaki, K.: Algorithmic information theory and fractal sets. In: Proceedings of 1999 Workshop on Information-Based Induction Sciences (IBIS 1999), Syuzenji, Shizuoka, Japan, August 26-27, 1999, pp. 105–110 (1999) (in Japanese)
13. Tadaki, K.: A generalization of Chaitin's halting probability Ω and halting self-similar sets. Hokkaido Math. J. 31, 219–253 (2002); Electronic Version, `http://arxiv.org/abs/nlin/0212001v1`
14. Tadaki, K.: A statistical mechanical interpretation of algorithmic information theory. In: Local Proceedings of Computability in Europe 2008 (CiE 2008), University of Athens, Greece, June 15-20, 2008, pp. 425–434 (2008); Extended and Electronic Version, `http://arxiv.org/abs/0801.4194v1`
15. Toda, M., Kubo, R., Saitô, N.: Statistical Physics I., 2nd edn. Equilibrium Statistical Mechanics. Springer, Berlin (1992)
16. Weihrauch, K.: Computable Analysis. Springer, Berlin (2000)

A The Proof of Theorem 3

For completeness, we prove here Theorem 3, based on Theorems 8, 10, and 11, in a similar manner to the proof of Theorem 4 given in Section 5.

Let $f\colon (0,1) \to \mathbb{R}$ with $f(T) = Z(T)$, and let $g\colon (0,1) \times \mathbb{N}^+ \to \mathbb{R}$ with $g(T,k) = Z_k(T)$. First, it follows that f is increasing.

Obviously, $\lim_{k\to\infty} g(T,k) = f(T)$ for every $T \in (0,1)$. It follows that, for every $T \in (0,1)$ and every $k \in \mathbb{N}^+$,

$$g(T,k+1) - g(T,k) = 2^{-\frac{|p_{k+1}|}{T}}.$$

Thus, for every $T \in (0,1)$ and every $k \in \mathbb{N}^+$, $g(T,k) < g(T,k+1)$ and therefore $g(T,k) < f(T)$. At this point, the conditions (i), (ii), and (iv) of Theorem 8, all conditions of Theorem 10, and the conditions (i), (ii), (iv), (v), and (vi) of Theorem 11 hold for g.

Using the mean value theorem we see that, for every $k \in \mathbb{N}^+$ and every $T, x, t \in (0,1)$ with $T < x < t$,

$$\frac{\ln 2}{t^2} W_k(T)(x-T) < g(x,k) - g(T,k) < \frac{\ln 2}{T^2} W_k(t)(x-T),$$

where $W_k(T) = \sum_{i=1}^{k} |p_i|\, 2^{-\frac{|p_i|}{T}}$, as defined before Lemma 1 in Section 4. Note that, for every $t \in (0,1)$, $W_k(t)$ is increasing on k, and $W(t) = \lim_{k\to\infty} W_k(t)$ exists by Lemma 1 (i). Thus we see that

$$\frac{\ln 2}{t^2} W_1(T)(x-T) < g(x,k) - g(T,k) < \frac{\ln 2}{T^2} W(t)(x-T),$$

for every $k \in \mathbb{N}^+$ and every $T, x, t \in (0,1)$ with $T < x < t$. Therefore, the condition (iii) of Theorem 8 and the condition (iii) of Theorem 11 hold for g.

Thus, Theorem 8, Theorem 10, and Theorem 11 result in the following three theorems, respectively.

Theorem 23. *For every $T \in (0,1)$, if $Z(T)$ is right-computable then T is weakly Chaitin T-random.* □

Theorem 24. *For every $T \in (0,1)$, if $Z(T)$ is right-computable then T is also right-computable.* □

Theorem 25. *For every $T \in (0,1)$, if $Z(T)$ is left-computable and T is right-computable, then T is T-compressible.* □

Theorem 3 follows immediately from these three theorems.

Decidability and Undecidability in Probability Logic

Sebastiaan A. Terwijn*

Institute of Logic, Language and Computation
University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands

Abstract. We study computational aspects of a probabilistic logic based on a well-known model of induction by Valiant. We prove that for this paraconsistent logic the set of valid formulas is undecidable.

1 Introduction

The probabilistic interpretation of quantifiers has a long tradition and has been studied in many forms, often motivated by the difficulties of obtaining a complete picture of the world outside the realm of mathematical formalisms. We will not attempt to give an historical overview of the various approaches in the restricted context of this brief paper, but instead confine the discussion to those sources that are of direct relevance to it. More references to papers concerning probability logic can be found in [6].

Valiant [10] and Terwijn [7] gave probabilistic interpretations of first-order predicate logic based on Valiants model of pac-learning. In these interpretations universal quantification in a model $\mathcal{M}$ is interpreted as "many", where "many" refers to a given probability distribution $\mathcal{D}$ on $\mathcal{M}$ and to a given error parameter ε. These probabilistic interpretations were partly motivated by considerations from computational learning theory. In this paper our concern is not the induction of formulas but the study of probabilistic truth itself. Both [10] and [7] are predated by Keisler [5] (that also surveys many results of other researchers, notably Hoover), in which a logic is studied with essentially the same probabilistic interpretation of universal quantification, but with no other quantifiers, and with a negation that is different from the one below. Our different interpretation of negation allows for having the classical existential quantifier $\exists$ in the logic, something that Keislers logic does not have. A logic with a measure quantifier was also introduced by H. Friedman (cf. Steinhorn in [1]), but this logic is even less related to ours than Keislers. A recent study of a probabilistic logic extending classical predicate logic that is motivated by inductive probabilistic reasoning is Jaeger [4].

* This research was supported by the Austrian Science Fund FWF under project P20346-N18.

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 441–450, 2009.

We start by repeating the definition of probabilistic truth from [7]. For unexplained measure-theoretic terminology we refer to Doob [3]. Fix a first-order language with predicates and constants, possibly with equality, but no function symbols. Let $\mathcal{M}$ be a classical model (consisting of a universe with interpretations of the predicates in the language) and let $\mathcal{D}$ be a probability measure on $\mathcal{M}$, i.e. a measure such that $\mathcal{D}(\mathcal{M}) = 1$. For a $\mathcal{D}$-measurable subset $X \subseteq \mathcal{M}$ we will sometimes write $\Pr_{\mathcal{D}}[X]$ instead of $\mathcal{D}(X)$, to emphasize that we think of these measures as probabilities. For the moment we just assume that $\mathcal{D}$ is a probability measure. We will discuss an additional property that one can impose on $\mathcal{D}$ in Section 2.

Given a property $\varphi(x)$ for elements in a model $\mathcal{M}$, and given an error parameter ε, one can calculate (using Chernoff bounds, cf. [7]) how large a sample of x's from $\mathcal{M}$ should be to be able to assert with a certain confidence $1-\delta$ that at least $1-\varepsilon$ of the x's in $\mathcal{M}$ (in terms of the unknown measure $\mathcal{D}$) satisfy $\varphi(x)$. In the context of such large samples, we want $\forall x\varphi(x)$ to mean that this is the case, i.e. that at least $1-\varepsilon$ of the x's in $\mathcal{M}$ satisfy $\varphi(x)$. In contrast, we want $\exists x\varphi(x)$ to mean that an x was found in the sample that satisfies $\varphi(x)$. $\neg\exists x\varphi(x)$ should mean that no such x was found, which is the same as saying that $\forall x\neg\varphi(x)$. $\neg\forall x\varphi(x)$ means that not all sampled x's satisfy $\varphi(x)$, that is, the sample contains an x with $\neg\varphi(x)$, i.e. $\exists x\neg\varphi(x)$. These considerations are reflected in the following definition. Note that we do not model induction of formulas here; at this point we are solely interested in probabilistic truth.

Definition 1.1. (Truth definition) Given $\varepsilon \in [0,1]$, we inductively define the relation $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$ as follows.

1. For every prime formula φ (i.e. φ atomic or the negation of an atomic formula), $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$ if $\mathcal{M} \models \varphi$.
2. The logical connectives $\wedge$ and $\vee$ are treated classically, e.g. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi \wedge \psi$ if it holds that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$ and $\mathcal{M} \models_{\mathcal{D},\varepsilon} \psi$.
3. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \exists x\varphi(x)$ if there exists $x \in \mathcal{M}$ such that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(x)$.
4. The case of negation is split into subcases as follows:
   4.1. For φ atomic, $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\neg\varphi$ if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$. Furthermore, $\neg$ distributes in the classical way over $\vee$ and $\wedge$, e.g. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg(\varphi\wedge\psi)$ if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\varphi\vee\neg\psi$.
   4.2. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\exists x\varphi(x)$ if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \forall x\neg\varphi(x)$.
   4.3. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\forall x\varphi(x)$ if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \exists x\neg\varphi(x)$.
5. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi \to \psi$ if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\varphi \vee \psi$.
6. $\mathcal{M} \models_{\mathcal{D},\varepsilon} \forall x\varphi(x)$ if $\Pr_{\mathcal{D}}[x \in \mathcal{M} : \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(x)] \geqslant 1-\varepsilon$.

Note that in the above definition everything is treated classically, except the interpretation of $\forall x\varphi(x)$ in Case 6. The treatment of negation requires some care, since we no longer have that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \neg\varphi$ implies that $\mathcal{M} \not\models_{\mathcal{D},\varepsilon} \varphi$ (though the converse still holds).

Note that both $\mathcal{M} \models_{\mathcal{D},\varepsilon} \exists x\varphi(x)$ and $\mathcal{M} \models_{\mathcal{D},\varepsilon} \forall x\neg\varphi(x)$ may hold, since the interpretation of the first is the classical one, but the interpretation of the second

is that *most* x's satisfy $\neg\varphi(x)$. That is, the logic of $\models_{\mathcal{D},\varepsilon}$ is *paraconsistent*. In [7] the above definition is motivated. In particular, the asymmetry in the interpretation of $\exists$ and $\forall$ is motivated by an interpretation in which the truth of first-order statements in an unknown model is established with a given degree of confidence by taking samples from the model.

Case 5 defines $A \to B$ as $\neg A \vee B$. We note that this is weaker than the classical implication. Namely, the classical definition would say that B holds in any model where A holds. Taking $B = \bot$, where $\bot$ is an inconsistency such as $\exists x(R(x) \wedge \neg R(x))$, we would thus obtain the classical negation of A. Taking for A an existential statement, then since $\exists$ expresses classical existence we would thus also obtain the classical universal quantifier $\forall$, and our logic would become a strong extension of classical predicate logic, which is not what we are after. Instead, $\exists x\varphi(x) \to \bot$ by definition means $\neg\exists x\varphi(x) \vee \bot$, which is the same as $\forall x\neg\varphi(x)$. Thus the above definition of implication takes on a probabilistic interpretation: If we interpret $\neg A$ by saying that A is *unlikely*, then $A \to B$ holds if whenever A holds it is *likely* that B holds.

Note that for $\varepsilon = 0$ the truth definition does not coincide with the classical one: If $\mathcal{M} \models_{\mathcal{D},0} \forall x R(x)$ there can still be a set of $\mathcal{D}$-measure zero of x's with $\neg R(x)$. *In the following we will exclude the pathological case of* $\varepsilon = 1$. Note that for $\varepsilon = 1$ all universal statements are always true, for example.

Proposition 1.1. (Prenex normal form) *Every formula φ is semantically equivalent to a formula φ' in prenex normal form, that is, φ' satisfies $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi \Leftrightarrow \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi'$ for all models $\mathcal{M}$, $\mathcal{D}$, ε.*

Proof. By Case 5 in Definition 1.1 we may assume that the formula is free of implications. Case 4 in the definition allows us to rewrite all formulas by pushing the negations inside, so that all negations occur only directly in front of an atomic formula. We then pull all quantifiers outside: Clearly we can pull $\exists$ outside over the connectives $\wedge$ and $\vee$ since $\exists$ has the classical meaning. For $\forall$ we have to check that

$$\varphi \wedge \forall x\psi(x) \equiv \forall x(\varphi \wedge \psi(x)) \tag{1}$$

and

$$\varphi \vee \forall x\psi(x) \equiv \forall x(\varphi \vee \psi(x)) \tag{2}$$

where $\equiv$ denotes semantic equivalence, and under the usual proviso about x not occurring free in φ. Indeed, for (1) we have

$$\begin{aligned}\mathcal{M} \models_{\mathcal{D},\varepsilon} \forall x\big(\varphi \wedge \psi(x)\big) &\iff \Pr_x\big[\varphi \wedge \psi(x)\big] \geqslant 1-\varepsilon \\ &\iff \varphi \wedge \Pr_x\big[\psi(x)\big] \geqslant 1-\varepsilon \\ &\iff \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi \wedge \forall x\psi(x).\end{aligned}$$

The second statement is proved in exactly the same way, replacing $\wedge$ by $\vee$. □

Definition 1.2. We will use the following terminology: By a *probabilistic model* we will mean a triple $\mathcal{M}, \mathcal{D}, \varepsilon$ as above, where $\varepsilon \in [0, 1)$. In this case we also call

the pair $\mathcal{M}$, $\mathcal{D}$ an *ε-model*. For a given ε, a sentence φ is *ε-satisfiable* if there are $\mathcal{M}$ and $\mathcal{D}$ such that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$, and φ is *ε-valid* if $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$ for every $\mathcal{M}$ and $\mathcal{D}$. Furthermore, φ is *probabilistically satisfiable* if φ is ε-satisfiable for some $\varepsilon < 1$.

Note that all models are necessarily nonempty since they are measure spaces. From Proposition 1.1 it is easy to see that for $\varepsilon \leqslant \varepsilon'$, every ε-valid formula is ε'-valid. See also Proposition 3.2 below. In Section 2 we will amend Definition 1.2 by imposing an extra restriction on the probabilistic models.

Definition 1.3. Below we will use the shorthand notation $\vec{z}$ for a series of variables $z_1, \ldots, z_n$. Let us adopt here the convention that for a formula $\varphi(\vec{z})$ with free variables $\vec{z}$ it holds that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(\vec{z})$ whenever there are $\vec{z} \in \mathcal{M}$ such that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(\vec{z})$. So we think of unbound variables as being existentially quantified.

2 The Measurability of Predicates

In Case 6 of Definition 1.1 we require in particular that the set

$$\big\{x \in \mathcal{M} : \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(x)\big\}$$

is $\mathcal{D}$-measurable. One can argue that it is natural to require a bit more than this, namely that

> for every k-ary predicate R occurring in φ the set of k-tuples satisfying R is $\mathcal{D}^k$-measurable, (3)

where $\mathcal{D}^k$ denotes the product measure on $\mathcal{M}^k$. This is a natural assumption: When we are talking about probabilities over certain predicates we may as well require that all such probabilities exist, even if in some cases this would not be necessary. The property (3) and its consequences are discussed more extensively in [8]. *Henceforth, we will assume property (3).*

3 The Set of 0-Valid Formulas

In this section we make some preliminary remarks about the set of 0-valid formulas. We start by repeating an easy preliminary result from [7].

Lemma 3.1. *Let $\mathcal{D}$ be a probability distribution on $\mathcal{M}$ such that for all $x \in \mathcal{M}$, $\mathcal{D}(\{x\}) \neq 0$. Then for every formula φ, $\mathcal{M} \models \varphi \iff \mathcal{M} \models_{\mathcal{D},0} \varphi$.*

Proof. One direction follows from the fact that classical validity implies probabilistic validity, since the only difference is that the probabilistic interpretation of $\forall$ is weaker. For the converse direction, if $\mathcal{D}$ is as in the lemma and $\Pr_{\mathcal{D}}\big[x \in \mathcal{M} : \mathcal{M} \models_{\mathcal{D},0} \varphi(x)\big] = 1$ then in fact $(\forall x \in \mathcal{M})\big[\mathcal{M} \models_{\mathcal{D},0} \varphi\big]$. So the interpretation of $\forall$ is in fact the classical one, and hence every formula is interpreted classically. □

Proposition 3.1. *The* 0*-valid formulas coincide with the classically valid formulas.*

Proof. That every classically valid formula is also probabilistically valid was already noted above. For the converse, suppose that φ is not classically valid. Then there is a countable model $\mathcal{M}$ such that $\mathcal{M} \not\models \varphi$. Since $\mathcal{M}$ is countable, there is a distribution $\mathcal{D}$ on $\mathcal{M}$ such that for all $x \in \mathcal{M}$, $\mathcal{D}(\{x\}) \neq 0$. But then by Lemma 3.1, $\mathcal{M} \not\models_{\mathcal{D},0} \varphi$. Hence φ is not 0-valid. □

Note however that we do not have that every 0-satisfiable sentence is classically satisfiable; a counterexample is $\exists x R(x) \wedge \forall x \neg R(x)$.

Proposition 3.2. (Terwijn [7]) *For all $\varepsilon < \varepsilon'$, the set of ε-valid formulas is strictly included in the set of ε'-valid ones.*[1]

Proposition 3.3. *Let $\varphi(\vec{x})$ be a formula with free variables $\vec{x}$ such that for every probabilistic model $\mathcal{M}$, $\mathcal{D}$ and every $\vec{x} \in \mathcal{M}$*

$$\forall \varepsilon > 0 \big(\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(\vec{x})\big) \Longrightarrow \mathcal{M} \models_{\mathcal{D},0} \varphi(\vec{x}). \tag{4}$$

If furthermore $\forall \vec{x} \varphi(\vec{x})$ is ε-valid for every $\varepsilon > 0$, then $\forall \vec{x} \varphi(\vec{x})$ is 0*-valid.*

Proof. By induction on the number of $\forall$-quantifiers it suffices to prove this for $\forall x \varphi(x)$, where $\varphi(x)$ satisfies (4). So suppose $\varphi(x)$ satisfies (4). Then

$$\begin{aligned}
\forall \varepsilon > 0 \;\; \mathcal{M} \models_{\mathcal{D},\varepsilon} \forall x \varphi(x) &\Longrightarrow \forall \varepsilon > 0 \;\; \Pr_{\mathcal{D}}\big[x : \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(x)\big] \geqslant 1 - \varepsilon \\
&\Longrightarrow \forall \varepsilon > 0 \;\; \Pr_{\mathcal{D}}\big[x : \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi(x)\big] = 1 \\
&\Longrightarrow \Pr_{\mathcal{D}}\Big(\bigcap_{n=2}^{\infty} \big\{x : \mathcal{M} \models_{\mathcal{D},\frac{1}{n}} \varphi(x)\big\}\Big) = 1 \\
&\Longrightarrow \Pr_{\mathcal{D}}\big[x : \mathcal{M} \models_{\mathcal{D},0} \varphi(x)\big] = 1 \\
&\Longrightarrow \mathcal{M} \models_{\mathcal{D},0} \forall x \varphi(x).
\end{aligned}$$

Here the second to last implication follows because φ satisfies (4). So if for every $\varepsilon > 0$ the sentence $\forall x \varphi(x)$ is ε-valid then it is 0-valid. □

Following standard notation, let $\forall^n \exists^m$ denote the class of $\mathcal{L}$-sentences in prenex form with at most n $\forall$-quantifiers followed by at most m $\exists$-quantifiers. Similarly, let $\exists^*$ and $\forall^*$ denote the fragments defined by any finite number of $\exists$ or $\forall$ quantifiers. Note that in contrast to the classical case, under the probabilistic interpretation we do *not* have that for example the $\forall^2$-fragment is closed under conjunctions. To see this, notice that the pair of formulas $\varphi_0 = \forall x \forall y Rxy \wedge$

[1] The proof in [7] actually does not take the extra measurability condition (3) into consideration. However an alternative proof using similar ideas of this result can be given that also respects (3).

$\forall x \forall y \neg Ryx$ and $\varphi_1 = \forall x \forall y (Rxy \wedge \neg Ryx)$ are not semantically equivalent. One can for example prove that whereas both formulas are $\frac{1}{3}$-satisfiable, φ_0 has a finite $\frac{1}{3}$-model, but φ_1 has not, cf. [8]. Hence, to put φ_0 in prenex form we have to rename variables and put more than two quantifiers in the prefix.

Corollary 3.1. *For every $\varphi \in \forall^* \exists^*$, if φ is not 0-valid then there is an $\varepsilon > 0$ such that φ is not ε-valid.*

Proof. It suffices to note that every formula φ of the form $\exists \vec{y} P(\vec{x}, \vec{y})$, where P is a propositional combination of atomic predicates, satisfies (4). This is because if $\mathcal{M} \models_{\mathcal{D},\varepsilon} P(\vec{x}, \vec{y})$ then $\mathcal{M} \models_{\mathcal{D},0} P(\vec{x}, \vec{y})$. □

At this point we do not know whether Corollary 3.1 holds for *all* sentences φ, i.e. whether $\bigcap_{\varepsilon > 0} \varepsilon$-valid = 0-valid.[2]

4 The Undecidability of the ε-Valid Formulas

In this section we prove that the set of ε-valid formulas is undecidable for every ε. Note that for $\varepsilon = 0$ the set of ε-valid formulas coincides with the classically valid formulas by Proposition 3.1, and hence is Σ^0_1-complete.

Definition 4.1. Given a probabilistic model $\mathcal{M}, \mathcal{D}$ and a subset $X \subseteq \mathcal{M}$ with $\mathcal{D}(X) > 0$, we define the *restriction* of $\mathcal{M}$ to X, denoted by $\mathcal{M}{\restriction}X$, as the model with universe X obtained from $\mathcal{M}$ by restricting all relations to X, and with the probability distribution on X defined by multiplying $\mathcal{D}$ with a factor $1/\mathcal{D}(X)$.

Theorem 4.1. *For every rational $\varepsilon \in [0, 1)$, the set of ε-valid formulas is Σ^0_1-hard.*

Proof. Suppose that $0 < m < n$ and that $\varepsilon = 1 - \frac{m}{n}$. We build a many-one reduction from the 0-valid formulas to the ε-valid ones, i.e. we show that there is a computable function f such that φ is 0-valid if and only if $f(\varphi)$ is ε-valid. Note that this suffices since by Proposition 3.1 the 0-valid formulas coincide with the classically valid ones, and these are of course undecidable.

The idea of the proof is to introduce new parts $X_0, \ldots X_{n-1}$ into a given model to "dilute" the meaning of the $\forall$-quantifiers in φ. We consider suitably relativized versions $\varphi^{X_{i_1} \cdots X_{i_m}}$ of φ relative to fractions $X_{i_1}, \ldots, X_{i_m}$ of m out of the n X_i's. In $\varphi^{X_{i_1} \cdots X_{i_m}}$ the existential quantifiers range over X_{i_1} and $X_{i_2}, \ldots, X_{i_m}$ are used to dilute the $\forall$ quantifiers in such a way that φ holds in X_{i_1} with error 0 if and only if $\varphi^{X_{i_1} \cdots X_{i_m}}$ holds in $X_{i_1} \cup \ldots \cup X_{i_m}$ with error 0. If $X_{i_1} \cup \ldots \cup X_{i_m}$ has weight $\geqslant \frac{m}{n}$ then the latter holds if and only if $\varphi^{X_{i_1} \cdots X_{i_m}}$ holds in $X_0 \cup \ldots \cup X_{n-1}$ with error at most ε. The main problem is to express all this correctly in such

[2] Note however that the same proof will not work, since in general (4) fails for $\exists\forall$-formulas. Corollary 3.1 was used in an earlier version of the proof of Theorem 4.1, in which a reduction was built from the formulas in the $\forall^3\exists$-fragment. This fragment is undecidable by a result of Surányi [2, Theorem 3.1.16]

a way that a 0-countermodel to φ can be transformed to ε-countermodel of the relativized version. The proof below would be would be considerably simpler if we could express $\mathcal{D}(X) > \frac{m}{n}$. To circumvent this technical difficulty, we resort to considering all possible combinations $i_1 \dots i_m$.

Formally, given a first-order formula φ, let $X_0, \dots, X_{n-1}$ be fresh unary predicates, i.e. predicates not occurring in φ. Define the sentence

$$n\text{-split} = \forall x \Big(\big(X_0(x) \vee \dots \vee X_{n-1}(x)\big) \wedge \bigwedge_{i<n} \Big[X_i(x) \leftrightarrow \bigwedge_{j \neq i} \neg X_j(x) \Big] \Big)$$

that says that $\mathcal{M}$ splits into n parts. Note that since n-split is purely universal we have that $\mathcal{M}$ probabilistically satisfies $\neg n$-split if and only if it satisfies it classically, hence if $\mathcal{M} \not\models_{\mathcal{D},\varepsilon} \neg n$-split then really $\mathcal{M} \models n$-split classically. In the following we use set-theoretic notation such as $x \in X_0 \cup \dots \cup X_{n-1}$ as a shorthand for the formula $X_0(x) \vee \dots \vee X_{n-1}(x)$. We also write $\mathcal{D}(X) \geqslant \frac{m}{n}$ for the sentence $\forall x(X(x))$. (Note that since $\varepsilon = 1 - \frac{m}{n}$ this last sentence expresses precisely this fact.) Given a sentence φ and a choice $i_1, \dots, i_m$ of m different numbers from the set $\{0, \dots, n-1\}$, define a relativized version $\varphi^{X_{i_1} \dots X_{i_m}}$ of φ by recursively replacing $\exists x$ everywhere by $\exists x(X_{i_1}(x) \wedge \dots)$ and all occurrences of $\forall x$ by $\forall x\big(x \in X_{i_2} \cup \dots \cup X_{i_m} \vee (x \in X_{i_1} \wedge \dots)\big)$.[3] For every φ define

$$f(\varphi) = \neg n\text{-split} \vee \bigvee_{i_1 \dots i_m} \Big(\mathcal{D}(X_{i_1} \cup \dots \cup X_{i_m}) \geqslant \tfrac{m}{n} \wedge \varphi^{X_{i_1} \dots X_{i_m}} \Big).$$

Here the disjunction is over all choices $i_1 \dots i_m$ of m different numbers from the set $\{0, \dots, n-1\}$. Now if φ is 0-valid and $\mathcal{M} \not\models_{\mathcal{D},\varepsilon} \neg n$-split then $\mathcal{M}$ splits into $X_0 \dots X_{n-1}$. By Lemma 4.1 there is always a choice of $i_1 \dots i_m$ such that $\mathcal{D}(X_{i_1} \cup \dots \cup X_{i_m}) \geqslant \frac{m}{n}$. Without loss of generality $\mathcal{D}(X_{i_1}) > 0$, for if this does not hold we can permute $i_1 \dots i_m$. But then by Lemma 4.2 we have that $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi^{X_{i_1} \dots X_{i_m}}$. Hence $f(\varphi)$ is ε-valid.

Conversely, suppose that φ is not 0-valid, say that $\mathcal{M} \not\models_{\mathcal{D},0} \varphi$. We show that there is a model $\mathcal{M}'$,$\mathcal{D}'$ such that $\mathcal{M}' \not\models_{\mathcal{D}',\varepsilon} f(\varphi)$. Let $\mathcal{M}'$ consist of the n disjoint parts $X_0, \dots, X_{n-1}$, where each X_i is a copy of $\mathcal{M}$ where in addition every element satisfies the unary predicate X_i. The predicates on $\mathcal{M}'$ are defined exactly as in $\mathcal{M}$ within each given copy X_i, and are defined arbitrarily across different copies. Under $\mathcal{D}'$ we give each of $X_0 \dots X_{n-1}$ weight $\frac{1}{n}$. The structure of $\mathcal{D}'$ on each X_i is like $\mathcal{D}$ on $\mathcal{M}$, multiplied with the factor $\frac{1}{n}$, that is, $\mathcal{D}'$ is the sum of n copies of $\frac{1}{n} \cdot \mathcal{D}$. Notice that by definition $\mathcal{M}'$ does not ε-satisfy $\neg n$-split, and that it ε-satisfies $\mathcal{D}(X_{i_1} \cup \dots \cup X_{i_m}) \geqslant \frac{m}{n}$ for any choice $i_1 \dots i_m$ of m different numbers from $\{0, \dots, n-1\}$. Given any such choice $i_1 \dots i_m$, let $\mathcal{M}' \restriction X_{i_1} \cup \dots \cup X_{i_m}$, $\mathcal{D}''$ be the restriction of $\mathcal{M}'$ to $X_{i_1} \cup \dots \cup X_{i_m}$. (Cf. Definition 4.1.) So $\mathcal{D}''$ on $X_{i_1} \cup \dots \cup X_{i_m}$ is $\mathcal{D}'$ multiplied with $1/\mathcal{D}'(X_{i_1} \cup \dots \cup X_{i_m}) = \frac{n}{m}$. Now suppose that $\mathcal{M}' \models_{\mathcal{D}',\varepsilon} \varphi^{X_{i_1} \dots X_{i_m}}$. Then by Lemma 4.3, $\mathcal{M}' \restriction X_{i_1} \cup \dots \cup X_{i_m} \models_{\mathcal{D}'',0} \varphi^{X_{i_1} \dots X_{i_m}}$. But since X_{i_1} is a copy of $\mathcal{M}$, this easily implies $\mathcal{M} \models_{\mathcal{D},0} \varphi$: By definition of $\varphi^{X_{i_1} \dots X_{i_m}}$, witnesses for existential quantifiers can be found in X_{i_1}, and universal

[3] If $m = 1$ then we replace $\forall x$ by $\forall x(x \in X_{i_1} \wedge \dots)$.

quantifiers hold with $\mathcal{D}''$-measure 1 in $\mathcal{M}'\restriction X_{i_1} \cup \ldots \cup X_{i_m}$, hence also with $\mathcal{D}$-measure 1 in X_{i_1}. Thus we have $\mathcal{M} \models_{\mathcal{D},0} \varphi$, contrary to the assumption. Hence $\mathcal{M}'$ also does not ε-satisfy $\varphi^{X_{i_1}\ldots X_{i_m}}$, and thus $\mathcal{M}'$ witnesses that $f(\varphi)$ is not ε-valid. □

Lemma 4.1. *Suppose that $n \geqslant m \geqslant 1$ and that $a_i \in \mathbb{R}$ are such that*

$$\sum_{i=0}^{n-1} a_i = 1.$$

Then there are m a_i's such that their sum is greater than or equal to $\frac{m}{n}$.

Proof. The average over the a_i is $\frac{1}{n}$, so the m largest of them sum up to at least $\frac{m}{n}$. □

Lemma 4.2. *In the proof of Theorem 4.1, suppose that $\mathcal{M} \not\models_{\mathcal{D},\varepsilon} \neg n\text{-}\mathsf{split}$, that $\mathcal{D}(X_{i_1} \cup \ldots \cup X_{i_m}) \geqslant \frac{m}{n}$, and that $\mathcal{D}(X_{i_1}) > 0$. Then $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi^{X_{i_1}\ldots X_{i_m}}$.*

Proof. This follows because φ is 0-valid, hence it holds with error 0 in $\mathcal{M}\restriction X_{i_1}$, which is defined since $\mathcal{D}(X_{i_1}) > 0$ (cf. Definition 4.1). Loosely speaking, $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi^{X_{i_1}\ldots X_{i_m}}$ holds because witnesses for the existential quantifiers can be found in X_{i_1} since φ holds in $\mathcal{M}\restriction X_{i_1}$, and the universal quantifiers in $\varphi^{X_{i_1}\ldots X_{i_m}}$ are satisfied since $\mathcal{D}(X_{i_1} \cup \ldots \cup X_{i_m}) \geqslant \frac{m}{n}$ and the error on X_{i_1} is 0. More formally, the lemma follows from the following claim. Let $\mathcal{D}'$ denote the distribution on $\mathcal{M}\restriction X_{i_1}$.
Claim. If $\mathcal{M}\restriction X_{i_1} \models_{\mathcal{D}',0} \varphi$ then $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi^{X_{i_1}\ldots X_{i_m}}$. The claim is proved by formula induction. By Proposition 1.1 we may assume that φ is in prenex normal form and that all negations occur directly in front of atomic predicates. The induction step for $\exists$ is trivial by definition of $\varphi^{X_{i_1}\ldots X_{i_m}}$, so the only case that requires attention is the induction step for $\forall$. So suppose that $\varphi = \forall x \psi(x)$. Then

$$\varphi^{X_{i_1}\ldots X_{i_m}} = \forall x\big(x \in X_{i_2} \cup \ldots \cup X_{i_m} \vee \big(x \in X_{i_1} \wedge \psi(x)^{X_{i_1}\ldots X_{i_m}}\big)\big). \qquad (5)$$

If $\mathcal{M}\restriction X_{i_1} \models_{\mathcal{D}',0} \varphi$ then

$$\Pr_{\mathcal{D}'}\big[x \in X_{i_1} : \mathcal{M}\restriction X_{i_1} \models_{\mathcal{D}',0} \psi(x)\big] \geqslant 1.$$

By induction hypothesis, for every $x \in X_{i_1}$ with $\mathcal{M}\restriction X_{i_1} \models_{\mathcal{D}',0} \psi(x)$ we have $\mathcal{M} \models_{\mathcal{D},\varepsilon} \psi(x)^{X_{i_1}\ldots X_{i_m}}$. It then follows from $\mathcal{D}(X_{i_1} \cup \ldots \cup X_{i_m}) \geqslant \frac{m}{n}$ that

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M} : x \in X_{i_2} \cup \ldots \cup X_{i_m} \vee \big(x \in X_{i_1} \wedge \psi(x)^{X_{i_1}\ldots X_{i_m}}\big)\big] \geqslant \tfrac{m}{n} = 1 - \varepsilon$$

hence $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi^{X_{i_1}\ldots X_{i_m}}$. □

Lemma 4.3. *In the proof of Theorem 4.1 we have that*

$$\mathcal{M}' \models_{\mathcal{D}',\varepsilon} \varphi^{X_{i_1}\ldots X_{i_m}} \Longrightarrow \mathcal{M}'\restriction X_{i_1} \cup \ldots \cup X_{i_m} \models_{\mathcal{D}'',0} \varphi^{X_{i_1}\ldots X_{i_m}}.$$

Proof. Again we prove this by induction on φ. By Proposition 1.1 we may assume that φ is in prenex normal form and that all negations occur directly in front of atomic predicates. Then all the steps in the induction are trivial, except the case of universal quantification. So suppose that $\varphi = \forall x \psi(x)$. Then $\varphi^{X_{i_1} \dots X_{i_m}}$ is of the form (5). Denoting $\psi(x)^{X_{i_1} \dots X_{i_m}}$ by $\hat{\psi}(x)$ we then have

$$\mathcal{M}' \models_{\mathcal{D}',\varepsilon} \varphi^{X_{i_1} \dots X_{i_m}} \quad \Longrightarrow$$

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M}' : x \in X_{i_2} \cup \dots \cup X_{i_m} \vee \big(x \in X_{i_1} \wedge \mathcal{M}' \models_{\mathcal{D}',\varepsilon} \hat{\psi}(x)\big)\big] \geqslant \tfrac{m}{n} \quad \Longrightarrow$$

$$\Pr_{\mathcal{D}}\big[x \in \mathcal{M}' : x \in X_{i_2} \cup \dots \cup X_{i_m} \vee \big(x \in X_{i_1} \wedge \mathcal{M}' \restriction X_{i_1} \cup \dots \cup X_{i_m} \models_{\mathcal{D}'',0} \hat{\psi}(x)\big)\big] \geqslant \tfrac{m}{n} \quad \Longrightarrow$$

$$\Pr_{\mathcal{D}'}\big[x \in X_{i_1} \cup \dots \cup X_{i_m} : x \in X_{i_2} \cup \dots \cup X_{i_m} \vee \big(x \in X_{i_1} \wedge \mathcal{M}' \restriction X_{i_1} \cup \dots \cup X_{i_m} \models_{\mathcal{D}'',0} \hat{\psi}(x)\big)\big] \geqslant \frac{\frac{m}{n}}{\mathcal{D}(X_{i_1} \cup \dots \cup X_{i_m})} = 1$$

$$\Longrightarrow \quad \mathcal{M}' \restriction X_{i_1} \cup \dots \cup X_{i_m} \models_{\mathcal{D}'',0} \varphi^{X_{i_1} \dots X_{i_m}}.$$

Here the second implication follows by the induction hypothesis. □

5 Finite Models and Decidability

It is shown in [8] that the downward Löwenheim-Skolem theorem fails for ε-logic: Not every infinitely ε-satisfiable sentence has a countable model. The next result shows that countable probabilistic models are in a way analogous to classical finite models:

Theorem 5.1. *Let φ be a sentence. Then*

$$\forall \mathcal{M} \text{ finite } \mathcal{M} \models \varphi \iff \forall \mathcal{M} \text{ countable} \forall \mathcal{D} \forall \varepsilon > 0 \ \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi.$$

Proof. (If) If $\mathcal{M}$ is finite and $\forall \varepsilon > 0 \ \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$ then classically $\mathcal{M} \models \varphi$: If $\mathcal{M}$ has n elements then take $\mathcal{D}$ the uniform distribution on $\mathcal{M}$ assigning to every element probability $\frac{1}{n}$ and take $\varepsilon < \frac{1}{n}$. Then there can be no exceptions to $\forall$-statements.

(Only if) The idea is simply that if $\mathcal{M}$ is countable then most of the weight under $\mathcal{D}$ is concentrated on *finitely many* elements of $\mathcal{M}$. If φ holds classically in all finite models, φ also holds on these finitely many elements. More precisely; Fix $\varepsilon > 0$ and a countable probabilistic model $\mathcal{M}$, $\mathcal{D}$, and suppose that φ is classically valid on all finite models. Let $\mathcal{M}' \subseteq \mathcal{M}$ be finite such that $\mathcal{M}'$ has weight at least $1 - \varepsilon$ under $\mathcal{D}$. Since $\mathcal{M}'$ is finite we have $\mathcal{M}' \models \varphi$. But then also $\mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi$, since clearly all existential quantifications from φ are satisfied within $\mathcal{M}$, and all universal quantifications have at most ε exceptions in weight. □

Notice that it is essential that in Theorem 5.1 we exclude the case $\varepsilon = 0$, since otherwise by Lemma 3.1 we would obtain all classical validities instead of only the finitely valid sentences.

Corollary 5.1. *The set*

$$\{\varphi : \forall \mathcal{M}\,\text{countable}\,\forall \mathcal{D}\forall \varepsilon > 0\ \ \mathcal{M} \models_{\mathcal{D},\varepsilon} \varphi\}$$

is Π_1^0-complete.

Proof. By Trakhtenbrot's theorem [9] (a result that was independently obtained by Craig) the set $\{\varphi : \forall \mathcal{M}\,\text{finite}\ \ \mathcal{M} \models \varphi\}$ of finitely valid first-order sentences is Π_1^0-complete. □

In Terwijn [8] it is proven that for fixed ε we do not have the finite model property: There are ε-satisfiable sentences without a finite ε-model. (Cf. the examples quoted on page 446.) Nevertheless, we make the following

Conjecture 5.1. For rational $\varepsilon \in [0, 1)$, it is decidable whether φ is ε-satisfiable.

Note that a positive answer to Conjecture 5.1 does not contradict the undecidability from Theorem 4.1 because, in contrast to the classical case, under the probabilistic interpretation we do not have that φ is valid if and only if $\neg\varphi$ is not satisfiable, even if $\varepsilon = 0$. For example the sentence $\varphi = \exists x R(x) \wedge \forall x \neg R(x)$ is probabilistically satisfiable, but its negation $\neg\varphi$ is $\forall x \neg R(x) \vee \exists x R(x)$, which is even classically valid.

References

1. Barwise, J., Feferman, S. (eds.): Model-theoretic logics. Springer, Heidelberg (1985)
2. Börger, E., Grädel, E., Gurevich, Y.: The classical decision problem. Springer, Heidelberg (1997)
3. Doob, J.L.: Measure theory. Springer, Heidelberg (1994)
4. Jaeger, M.: A logic for inductive probabilistic reasoning. Synthese 144(2), 181–248 (2005)
5. Keisler, H.J.: Probability quantifiers. In: [1], pp. 509–556
6. Probability logic papers, database, `http://problog.mi.sanu.ac.yu/index.html`
7. Terwijn, S.A.: Probabilistic logic and induction. Journal of Logic and Computation 15(4), 507–515 (2005)
8. Terwijn, S.A.: Model-theoretic aspects of probability logic (in preparation)
9. Trakhtenbrot, B.: The impossibility of an algorithm for the decision problem for finite models. Dokl. Akad. Nauk SSSR 70, 572–596 (1950); English translation in AMS Transl. Ser. 2, 1–6 (1963)
10. Valiant, L.G.: Robust logics. Artificial Intelligence 117, 231–253 (2000)

A Bialgebraic Approach to Automata and Formal Language Theory

James Worthington

Mathematics Department, Cornell University
Ithaca, NY 14853-4201 USA
worthing@math.cornell.edu

Abstract. A bialgebra is a structure which is simultaneously an algebra and a coalgebra, such that the algebraic and coalgebraic parts are "compatible". In this paper, we apply the defining diagrams of algebras, coalgebras, and bialgebras to categories of semimodules and semimodule homomorphisms over a commutative semiring. We then show that formal language theory and the theory of bialgebras have essentially undergone "convergent evolution". For example, formal languages correspond to elements of dual algebras of coalgebras, automata are "pointed representation objects" of algebras, automaton morphisms are instances of linear intertwiners, and a construction from the theory of bialgebras shows how to run two automata in parallel. We also show how to associate an automaton with an arbitrary algebra, which in the classical case yields the automaton whose states are formal languages and whose transitions are given by language differentiation.

1 Introduction

Automata and formal languages are standard objects of study in theoretical computer science. Classically, they have been studied from the algebraic perspective, focusing on transition matrices of automata, algebraic operations defined on formal power series, etc., as in the Kleene-Schützenberger theorem. Recently, automata have been studied from a coalgebraic perspective, focusing on the co-operations of transition and observation, and the coalgebraic notion of bisimulation. See, for example, [16].

In this paper, we treat automata and formal languages from a *bialgebraic* perspective: one that includes both algebraic and coalgebraic structures, with appropriate interactions between the two. This provides a rich framework to study automata and formal languages. Using bialgebras, we can succinctly express operations on automata, operations on languages, maps between automata, language homomorphisms, derivatives of languages, and the interactions among them. This raises the possibility of using methods and results from bialgebra theory to study problems in the theory of automata and formal languages such as the problem of constructing a proof of the equivalence of two nondeterministic automata accepting the same language (see Section 10 for further discussion).

S. Artemov and A. Nerode (Eds.): LFCS 2009, LNCS 5407, pp. 451–467, 2009.

We proceed by defining a bialgebra B on the set of all finite words over an alphabet Σ. The algebraic operation of multiplication describes how to "put words together"; it is essentially concatenation. The coalgebraic operation of comultiplication, a map $B \to B \otimes B$, describes how to "split words apart"; there are several comultiplications of interest. Note that this comultiplication is not the co-operation on automata used in [16], and so provides another example of the utility of co-operations in computer science.

Given an algebra A, we are interested in the structures on which A acts, i.e., its *representation objects*. We can encode an automaton as a representation object of A equipped with a start state and an observation function. These automata compute elements of the dual module of A, which we view as formal languages. Automaton morphisms, i.e., linear maps between automata which preserve the language accepted, are shown to be instances of *linear intertwiners*. Given a coalgebra C, the dual module of C also corresponds to a set of languages. A standard result is that a comultiplication on a coalgebra C defines a multiplication on the dual module; i.e., languages can be multiplied. If a structure is a bialgebra, these two views of formal languages interact nicely, and we can use a bialgebraic construction to "run two automata in parallel." In the other direction, given an algebra A, we show how to associate an automaton to A in a natural way. In the classical case, this corresponds to an automaton with formal languages for states and transitions given by Brzozowski derivatives.

Other authors have explored the role of bialgebras in the theory of automata and formal languages. In [8] and [9], Grossman and Larson study the question of which elements of the dual of a bialgebra can be represented by the action of the bialgebra on a finite object and prove the Myhill-Nerode theorem using notions from the theory of algebras. Our definition of an automaton is a straightforward generalization of theirs. However, we expand the perspective of [8] and [9] from bialgebras over fields to bialgebras over semirings. This generalization allows (for example) nondeterministic automata and their automaton morphisms to be expressed in the language of bialgebras. In [4] and [5], Duchamp et al. examine rationality-preserving operations of languages defined using various comultiplications on the algebra of input words, and construct the corresponding automata. They also apply these ideas to problems in combinatorial physics.

This paper is organized as follows. In Section 2, we define algebras, coalgebras, and bialgebras over a commutative ring R. In Section 3, we give the definitions of semirings and semimodules, and recall some useful facts and constructions. Section 4 contains the definition of the tensor product of two semimodules over a commutative semiring. Using this definition, in Section 5 we explain how to apply the defining diagrams of algebras, coalgebras, and bialgebras to categories of semimodules and semimodule homomorphisms over a commutative semiring. In Section 6, we use algebras to define automata, and in Section 7, we explore the relation between coalgebras of words and algebras of formal languages. In Section 8, we discuss bialgebras, tensor products of automata, and convolution products of languages. Section 9 contains a demonstration of how to endow an arbitrary

algebra with an automaton structure. The conclusion and future directions are in Section 10.

2 Algebras, Coalgebras, and Bialgebras

In this section, we define algebras, coalgebras, and bialgebras over a commutative ring R. This material is completely standard; see [15] or [17] (note that a Hopf algebra/quantum group is a special case of a bialgebra).

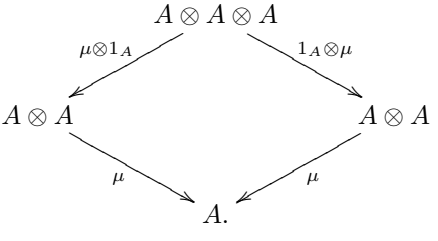
2.1 Algebras

We recall the definition of an R-algebra.

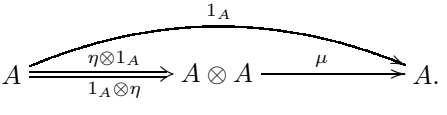
Definition 2.1. *Let R be a commutative ring. An* R-algebra *A is a ring A together with an injection $\eta : R \to A$ such that $\eta(R)$ is contained in the center of A and $\eta(1_R) = 1_A$.*

Remark 2.1. The function η is called the *unit map*. It is frequently defined as an arbitrary ring homomorphism $R \to A$. Since we require η to be an injection, we abuse notation and treat R as a subset of A.

To define an R-algebra diagrammatically, consider A as an R-module. Multiplication in A is an R-bilinear map $A \times A \to A$, by distributivity and the fact that R is contained in the center of A. By the universal property of the tensor product, multiplication defines a unique R-linear map $\mu : A \otimes A \to A$ (all tensor products in this section are over R). Associativity of multiplication means that the following diagram commutes:



The properties of the unit map can be expressed by the following commutative diagram (Recall that $A \otimes R \cong A \cong R \otimes A$):



Hence the diagrammatic definition of an R-algebra is an R-module A together with R-module homomorphisms $\mu : A \otimes A \to A$ and $\eta : R \to A$ such that the above diagrams commute.

Example 2.1. Let K be a field, and let x, y be indeterminates. Let A be the set of polynomials over noncommuting variables x, y with coefficients in K. Addition and multiplication of polynomials make A into a ring. To make A into an algebra, define $\eta(k)$ to be the constant polynomial $f(x, y) = k$ for $k \in K$.

Structure-preserving maps between algebras are called algebra maps.

Definition 2.2. *Let A and B be R-algebras. An* algebra map *is an R-linear map*
$f : A \to B$ *such that* $f(a_1 a_2) = f(a_1) f(a_2)$ *for all* $a_1, a_2 \in A$, *and* $f(1_A) = 1_B$.

Algebra maps can also be defined diagrammatically.

Definition 2.3. *Let A, B, be R-algebras. An* algebra morphism *is an R-linear map $f : A \to B$ such that*

$$\begin{array}{ccc} A \otimes A & \xrightarrow{f \otimes f} & B \otimes B \\ {\scriptstyle \mu_A}\downarrow & & \downarrow{\scriptstyle \mu_B} \\ A & \xrightarrow{f} & B \end{array} \qquad \begin{array}{ccc} & R & \\ {\scriptstyle \eta_A}\swarrow & & \searrow{\scriptstyle \eta_B} \\ A & \xrightarrow{f} & B. \end{array}$$

Given two algebras A and B, $A \otimes B$ becomes an algebra with multiplication

$$(a \otimes b) \cdot (a' \otimes b') = aa' \otimes bb'.$$

Diagrammatically, this multiplication can be expressed as a morphism

$$(A \otimes B) \otimes (A \otimes B) \xrightarrow[1_A \otimes \sigma \otimes 1_B]{\cong} (A \otimes A) \otimes (B \otimes B) \xrightarrow{\mu_A \otimes \mu_B} A \otimes B.$$

Here $\sigma : A \otimes B \to B \otimes A$; $\sigma(a \otimes b) = (b \otimes a)$ is the usual transposition map. The unit of $A \otimes B$ is given by

$$R \xrightarrow{\cong} R \otimes R \xrightarrow{\eta_A \otimes \eta_B} A \otimes B.$$

2.2 Coalgebras

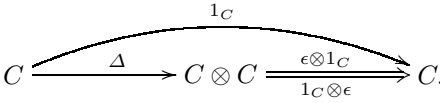
Dualizing the defining diagrams of an R-algebra yields an R-coalgebra.

Definition 2.4. *Let R be a commutative ring. An* R-coalgebra (C, Δ, ϵ) *is an R-module C and an R-linear coassociative function $\Delta : C \to C \otimes C$, called* comultiplication, *along with a linear* counit *map $\epsilon : C \to R$.*

Coassociativity of Δ means that the following diagram commutes:

$$\begin{array}{ccccc} & & C \otimes C \otimes C & & \\ & {\scriptstyle \Delta \otimes 1_C}\nearrow & & \nwarrow{\scriptstyle 1_C \otimes \Delta} & \\ C \otimes C & & & & C \otimes C \\ & {\scriptstyle \Delta}\nwarrow & & \nearrow{\scriptstyle \Delta} & \\ & & C. & & \end{array}$$

Diagrammatically, the axioms of the counit map are given by:



When performing calculations involving comultiplication, we often write

$$\Delta(c) = \sum_i c_{(1)} \otimes c_{(2)}$$

to express how c is "split" into elements of $C \otimes C$.

Example 2.2. Let P the set of polynomials over noncommuting variables x, y from Example 2.1. The map $\Delta : P \to P \otimes P$, defined on monomials w by $\Delta(w) = w \otimes w$ and extended linearly to all of P, is coassociative. The counit map $\epsilon : P \to R$ is evaluation at (1,1).

Coalgebras also have structure-preserving maps.

Definition 2.5. *Let C, D be R-coalgebras. A* coalgebra map *is an R-module homomorphism $g : C \to D$ such that $(g \otimes g) \circ \Delta_C = \Delta_D \circ g$ and $\epsilon_C = \epsilon_D \circ g$. The diagrams are the duals of the diagrams for algebra maps.*

Given coalgebras C and D, there is a natural coalgebra structure on $C \otimes D$. Comultiplication and counit are defined by

$$C \otimes D \xrightarrow{\Delta_C \otimes \Delta_D} (C \otimes C) \otimes (D \otimes D) \xrightarrow[1_C \otimes \sigma \otimes 1_D]{\cong} (C \otimes D) \otimes (C \otimes D).$$

$$C \otimes D \xrightarrow{\epsilon_C \otimes \epsilon_D} R \otimes R \cong R.$$

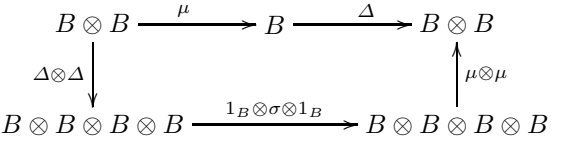
2.3 Bialgebras

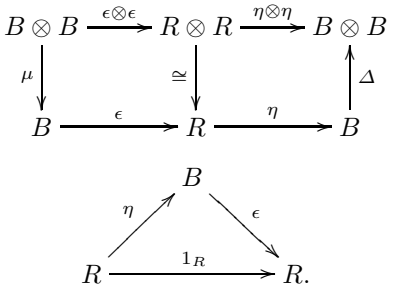
A R-bialgebra is an R-module which is both an R-algebra and an R-coalgebra, such that the two structures are compatible.

Definition 2.6. *Let R be a commutative ring. An* R-bialgebra *$(B, \mu, \eta, \Delta, \epsilon)$ is an R-module B which is a both an algebra and a coalgebra, satisfying:*

$$\Delta(ab) = \Delta(a)\Delta(b),\ \Delta(1) = 1 \otimes 1,\ \epsilon(ab) = \epsilon(a)\epsilon(b),\ \epsilon(1) = 1.$$

Note that the product $\Delta(a)\Delta(b)$ takes place in the algebra structure on $B \otimes B$. The defining diagrams for a bialgebra are as follows:





Remark 2.2. Note the "self-duality" of the defining diagrams of a bialgebra: swapping Δ for μ, ϵ for η, and reversing the direction of each arrow yields the same diagrams.

Example 2.3. Let M be a monoid and R a commutative ring. Let $R(M)$ be the free R-module on M. Define multiplication in $R(M)$ by extending multiplication in M linearly. This operation along with unit map $\eta(r) = r1_M$ forms an R-algebra structure on $R(M)$. There is an R-coalgebra structure on $R(M)$; define

$$\Delta(m) = m \otimes m$$

$$\epsilon(m) = 1$$

for $m \in M$ and extend linearly to $R(M)$. A straightforward calculation shows that this is an R-bialgebra structure on $R(M)$. Note that this includes Example 2.1 and 2.2 as a special case.

Finally, we give the definition of a bialgebra map.

Definition 2.7. *Let B, B' be bialgebras. A function $f : B \to B'$ is a* bialgebra map *if f is both an algebra map and a coalgebra map.*

3 Semirings and Semimodules

The above definition of a bialgebra is valid for any commutative ring R. However, in the theory of automata and formal languages, it is desirable to work over *semirings*, which are "rings without subtraction".

Definition 3.1. *A* semiring *is a structure $(K, +, \cdot, 0, 1)$ such that $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, and the following laws hold:*

$$a(b + c) = ab + ac$$

$$(b + c)a = ba + ca$$

$$0a = a0 = 0$$

for all $a, b, c \in K$. If $(K, \cdot, 1)$ is a commutative monoid, then K is said to be a commutative semiring. *If $(K, +, 0)$ is an idempotent monoid, then K is said to be an* idempotent semiring.

The representation objects of semirings are known as *semimodules*.

Definition 3.2. *Let K be a semiring. A* left K-semimodule *is a commutative monoid M along with a left action of K on M. The action satisfies the following axioms:*

$$(k_1 + k_2)m = k_1 m + k_2 m$$

$$k_1(m_1 + m_2) = k_1 m_1 + k_1 m2$$

$$(k_1 k_2)m = k_1(k_2 m)$$

$$1_K m = m$$

$$k_1 0_M = 0_M = 0_K m_1$$

for all $k_1, k_2 \in K$, $m_1, m_2 \in M$. If addition in M is idempotent, M is said to be an idempotent left K-semimodule.

Right K-semimodules are defined analogously. If K is commutative, then every left K-semimodule can be regarded as a right K-semimodule, and vice versa. In this case we omit the words "left" and "right".

Semimodules can be combined using the operations of direct sum and direct product.

Definition 3.3. *Let K be a commutative semiring and $\{M_i \mid i \in I\}$ be a collection of K-semimodules for some index set I. Let M be the cartesian product of the underlying sets. The* direct product *of the M_i's, denoted $\prod M_i$, is the set M endowed with pointwise addition and scalar multiplication. The* direct sum *of the M_i's, denoted $\bigoplus M_i$, is the subsemimodule of $\prod M_i$ in which all but finitely many of the coordinates are 0.*

Homomorphisms, congruence relations, free semimodules, and factor semimodules are all defined standardly. In the sequel, we will use elementary facts about them without comment. See [7] for definitions and proofs. We end this section a useful proposition about commutative semimodules.

Proposition 3.1. *Let K be a commutative semiring and M a K-semimodule. The set $Hom(M, K)$ of all K-linear maps from M to K is a K-semimodule.*

Proof. The usual proof for a module over a commutative ring is valid in this case. Commutativity of K is needed to show that the resulting functions are K-linear.

4 Tensor Products over Commutative Semirings

We wish to apply the defining diagrams of algebras, coalgebras, and bialgebras to categories of K-semimodules and K-linear maps for K a commutative semiring. To do this, we need a notion of the tensor product of K-semimodules. Unfortunately, the literature contains multiple inequivalent definitions of the tensor product of K-semimodules: the tensor product as defined in [7] is not the same

as the tensor product defined in [14] or [10]. In fact, the tensor product defined in [7] is the trivial K-semimodule when applied to idempotent K-semimodules.

We proceed by assuming that K is commutative and following the construction of the tensor product of modules over a commutative ring in [13]. This is essentially the construction used in [14] and [10]. The point is to work in the appropriate category and construct an object with the appropriate universal property.

We recall the universal property of the tensor product over a commutative ring R. Let $M_1, M_2, ..., M_n$ be R-modules. Let $\mathcal{C}$ be the category whose objects are n-multilinear maps

$$f : M_1 \times M_2 \times \cdots \times M_n \to F$$

where F ranges over all R-modules. To define the morphisms of $\mathcal{C}$, let

$$f : M_1 \times M_2 \times \cdots \times M_n \to F \text{ and } g : M_1 \times M_2 \times \cdots \times M_n \to G$$

be objects of $\mathcal{C}$. A morphism $f \to g$ is an R-linear map h such that $h \circ f = g$. A *tensor product* of $M_1, M_2, ..., M_n$, denoted $M_1 \otimes_R M_2 \otimes_R \cdots \otimes_R M_n$, is an initial object in this category. When it is clear from context, we omit the subscript on the $\otimes$ symbol. By standard category-theoretic arguments, the tensor product is unique up to isomorphism.

Let K be a commutative semiring and $M_1, M_2, ..., M_n$ be K-semimodules. Let T be the free K-semimodule on the set $M_1 \times M_2 \times \cdots \times M_n$. Let $\equiv$ be the congruence relation on T generated by the equivalences

$$(m_1, ..., m_i +_{M_i} m_i', ..., m_n) \equiv (m_1, ..., m_i, ..., m_n) +_T (m_1, ..., m_i', ..., m_n)$$

$$(m_1, ..., km_i, ..., m_n) \equiv k(m_1, ..., m_i, ..., m_n)$$

for all $k \in K, m_i, m_i' \in M_i, 1 \leq i \leq n$.

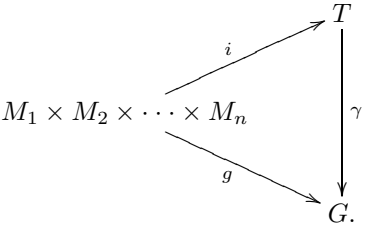
Let $i : M_1 \times M_2 \times \cdots \times M_n \to T$ be the canonical injection of $M_1 \times M_2 \times \cdots \times M_n$ into T. Let ϕ be the composition of i and the quotient map $q : T \to T/\equiv$.

Lemma 4.1. *The map ϕ is multilinear and is a tensor product of $M_1, M_2, ..., M_n$.*

Proof. Multilinearity of ϕ is obvious from its definition. Let G be a K-semimodule and

$$g : M_1 \times M_2 \times \cdots \times M_n \to G$$

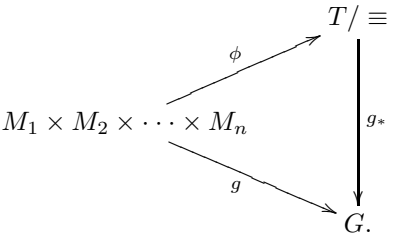
be a K-multilinear map. By freeness of T, there is an induced homomorphism (K-linear map) $\gamma : T \to G$ such that the following diagram commutes:



The homomorphism γ defines a congruence relation, denoted $\equiv_\gamma$, on T via

$$t \equiv_\gamma t' \text{ iff } \gamma(t) = \gamma(t')$$

for all $t, t' \in T$. Since g is K-multilinear, we have $\equiv \quad \subseteq \quad \equiv_\gamma$, where $\equiv$ is the congruence relation used in the definition of the tensor product. Therefore γ can be factored through $T/\equiv$, and there is a K-linear map $g_* : T/\equiv \quad \to G$ making the following diagram commute:



The image of ϕ generates $T/\equiv$, so g_* is uniquely determined.

For $x_i \in M_i$, we denote $\phi(x_1, x_2, ..., x_n)$ by $x_1 \otimes x_2 \otimes \cdots \otimes x_n$. Tensor products enjoy many useful properties.

Lemma 4.2. *Let K be a commutative semiring and $N, M_1, M_2, ..., M_n$ be K-semimodules. Then:*

1. *There is a unique isomorphism*

$$(M_1 \otimes M_2) \otimes M_3 \to M_1 \otimes (M_2 \otimes M_3)$$

   such that $(m_1 \otimes m_2) \otimes m_3 \mapsto m_1 \otimes (m_2 \otimes m_3)$ for all $m_i \in M_i$.
2. *There is a unique isomorphism $M_1 \otimes M_2 \to M_2 \otimes M_1$ such that $m_1 \otimes m_2 \mapsto m_2 \otimes m_1$ for all $m_i \in M_i$.*
3. *$K \otimes M_1 \cong M_1$*
4. *Let $\phi : M_1 \to M_3$ and $\psi : M_2 \to M_4$ be K-linear maps. There is a unique K-linear map $\phi \otimes \psi : M_1 \otimes M_2 \to M_3 \otimes M_4$ such that $(\phi \otimes \psi)(m_1 \otimes m_3) = \phi(m_1) \otimes \psi(m_2)$ for all $m_1 \in M_1, m_2 \in M_2$.*
5. *$N \otimes \bigoplus_{i=1}^n M_i \cong \bigoplus_{i=1}^n N \otimes M_i$*

Proof. In [13], these properties are proven for tensor products over commutative rings. The proofs rely on the universal property of the tensor product and are also valid in this case.

5 *K*-Algebras, *K*-Coalgebras, and *K*-Bialgebras

Let K be a commutative semiring. We define K-algebras, K-coalgebras, and K-bialgebras by applying the relevant diagrams from Section 2 to the category of K-semimodules and K-linear maps.

Example 5.1. Let $\Sigma = \{x, y\}$ be a set of noncommuting variables. Let P be the set of polynomials over Σ with coefficients from the two-element idempotent semiring K. Multiplication of polynomials is readily seen to be a bilinear function $P \times P \to P$, and therefore corresponds to a K-linear map $P \otimes_K P \to P$. Moreover, this map satisfies the associativity diagram. The map $\eta : K \to P$ such that $\eta(k) \mapsto \lambda xy.k$ satisfies the defining diagram of the unit map, and so P together with these maps forms a K-algebra. The map defined on monomials as $\Delta(w) = w \otimes w$ and extended linearly to all of P is easily seen to be coassociative. Defining $\epsilon(p(x, y)) = p(1, 1)$ makes P into a K-coalgebra. Furthermore, these maps satisfy the compatibility condition of a K-bialgebra, and so P is a K-bialgebra. We call constructions involving P "the classical case" and use P as an example throughout the sequel.

More generally, $R(G)$ from Example 2.3 with an underlying commutative semiring is a K-bialgebra.

6 *K*-Algebras and Automata

We use actions of K-algebras on K-semimodules to represent transitions of automata.

Definition 6.1. *Let A be a K-algebra and M be a K-semimodule. A* left action *of A on M is a K-linear map $A \otimes M \to M$, denoted $\triangleright$, satisfying*

$$(aa') \triangleright m = a \triangleright (a' \triangleright m)$$

$$1 \triangleright m = m$$

for all $a, a' \in A, m \in M$.

Right actions are defined analogously as K-linear maps $\triangleleft : M \otimes A \to M$. To define an automaton, we also need a start state and an observation function.

Definition 6.2. *A* (left) automaton $C = (M, A, s, \triangleright, \alpha)$ *consists of the following:*

1. *A K-algebra A, a K-semimodule M, and a left action $\triangleright$ of A on M.*
2. *An element $s \in M$, called the* start vector.
3. *A linear map $\alpha : M \to K$, called the* observation function.

That is, automata are "pointed observable representation objects" of a K-algebra A. Right automata are defined similarly using a right action $\triangleleft$. In the sequel, we will only give "one side" of a theorem or definition involving automata; the other follows mutatis mutandis.

Example 6.1. Consider the matrix representation of a classical two-state automaton with input alphabet $\{x, y\}$:

$$\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & x \\ y & 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right).$$

The leftmost matrix encodes the fact that the first state of the automaton is the start state. The 2×2-matrix encodes the transitions of the automaton, and the rightmost encodes the fact that the first state of the automaton is also the accept state. It is easy to see that this automaton accepts all words in $\{x, y\}^*$ of the form $(xy)^*$. Cf. automata encodings in [12]. We now translate this automaton to the framework of K-algebras.

Let K be the two-element idempotent semiring. Let S be the free K-semimodule on the set $\{s_1, s_2\}$, and let P be defined as in Example 5.1. Define a right action of the generators of P on S as follows:

$$\begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix} \triangleleft x = \begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix} \triangleleft y = \begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

and extend to an action of P on M. The start vector is

$$\begin{bmatrix} 1 & 0 \end{bmatrix}$$

and the observation function is

$$\alpha\left(\begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix}\right) = \begin{bmatrix} k_1 s_1 & k_2 s_2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Automata compute elements of $\mathrm{Hom}(A, K)$, as in [8].

Definition 6.3. *Let $C = (M, A, s, \triangleright, \alpha)$ be a left automaton. The* language accepted *by C is the function $\rho_C : A \to K$ such that*

$$\rho_C(a) = \alpha(a \triangleright s).$$

Lemma 6.1. *The function ρ_C as defined above is an element of $Hom(A, K)$.*

Proof. Immediate since $\triangleright$ and α are K-linear maps.

Much of the theory of automata concerns functions between automata which preserve the language accepted; these also have algebraic analogs.

Definition 6.4. *Let $C = (M, A, s_C, \triangleright_C, \alpha_C)$ and $D = (N, A, s_D, \triangleright_D, \alpha_D)$ be left automata over a K-algebra A. An* automaton morphism *from C to D is a map $\phi : M \to N$ such that*

$$\phi(s_C) = s_D \tag{1}$$

$$\phi(a \triangleright_C m) = a \triangleright_D \phi(m) \tag{2}$$

$$\alpha_C(m) = \alpha_D(\phi(m)) \tag{3}$$

for all $m \in M, n \in N, a \in A$.

Remark 6.1. Let V and W be modules. In the theory of algebras, a linear function $f : V \to W$ which satisfies (2) is known as a *linear intertwiner*. In the theory of automata, functions formally similar to automaton morphisms have been called *linear sequential morphisms* [1], *relational simulations* [2], *boolean bisimulations* [6], and *disimulations* [18]. Disimulations are based on the *bisimulation lemma* of Kleene algebra [12].

Algebra maps can be used to translate the input of an automaton.

Definition 6.5. *Let A, A' be K-algebras and $f : A \to A'$ a K-algebra map. Suppose A' acts on a K-semimodule M. Then A also acts on M according to the formula*

$$a \triangleright m = f(a) \triangleright m$$

for $a \in A, m \in M$. This is known as the pullback *of the action of A'.*

Automata theorists will recognize pullbacks as the main ingredient in the proof that regular languages are closed under inverse homomorphisms.

Let A be an arbitrary K-algebra. Elements of $\mathrm{Hom}(A, K)$ can be added since $\mathrm{Hom}(A, K)$ is a K-semimodule by Proposition 3.1. Given two automata C and D, there is an automaton accepting $\rho_C + \rho_D$.

Definition 6.6. *Let $C = (M, A, s_C, \triangleright_C, \alpha_C)$ and $D = (N, A, s_D, \triangleright_D, \alpha_D)$ be left automata over a K-algebra A. The* direct sum *of C and D is the automaton $C \oplus D = (M \oplus N, A, (s_C, s_D), \triangleright_{C\oplus D}, \alpha_C \oplus \alpha_D)$ where*

$$\triangleright_{C\oplus D} : A \otimes (M \oplus N) \to M \oplus N,$$

$$\triangleright_{C\oplus D}(a \otimes (m, n)) = ((a \triangleright_C m), (a \triangleright_D n))$$

and

$$\alpha_{C\oplus D} : M \oplus N \to K,$$

$$\alpha_{C\oplus D}(m, n) = \alpha_C(m) + \alpha_D(n).$$

The verification that $\triangleright_{C\oplus D}$ is an action of A on $M \oplus N$ is straightforward.

Theorem 6.1. *Let $C = (M, A, s_C, \triangleright_C, \alpha_C)$ and $(N, A, s_D, \triangleright_D, \alpha_D)$ be left automata over a K-algebra A. Then $\rho_{C\oplus D}(a) = \rho_C(a) + \rho_D(a)$ for all $a \in A$.*

Proof. Follows from the definitions:

$$\rho_{C\oplus D}(a) = \alpha_{C\oplus D}(a \triangleright_{C\oplus D} (s_C, s_D))$$

$$= \alpha_{C\oplus D}(a \triangleright_C s_C, a \triangleright_D s_D)$$

$$= \alpha_C(a \triangleright_C (s_C)) + \alpha_D(a \triangleright_D (s_D))$$

$$= \rho_C(a) + \rho_D(a).$$

7 K-Coalgebras and Formal Languages

Let C be a K-coalgebra. By Proposition 3.1, $\mathrm{Hom}(C, K)$ is a K-semimodule under the operations of pointwise addition and scalar multiplication. The coalgebra structure of C defines an algebra structure on $\mathrm{Hom}(C, K)$.

Definition 7.1. *Let (C, Δ, ϵ) be a K-coalgebra and $f, g \in Hom(C, K)$. The* convolution product *of f and g is defined by*

$$f * g = \mu_K \circ (f \otimes g) \circ \Delta.$$

Proposition 7.1. *Let $(C, +, \Delta, \epsilon)$ be a K-coalgebra. $Hom(C, K)$ is a K-algebra with associative multiplication given by the convolution product and unit*

$$\eta : K \to Hom(C, K)$$

$$\eta(k) = k\epsilon.$$

In particular, the multiplicative identity is ϵ_C.

Proof. The proof for coalgebras over a commutative ring suffices; it uses only the coassociativity of Δ and the universal property of the tensor product. See [15] or [17].

Example 7.1. Let P be as in Example 5.1. Note that an element of $\mathrm{Hom}(P, K)$ is completely determined by its values on monomials, which can be viewed as words in $\{x, y\}^*$. Thus there is a one-to-one correspondence between subsets of $\{x, y\}^*$ and elements of $\mathrm{Hom}(P, K)$.

The comultiplication defined on monomials as $\Delta(w) = w \otimes w$ and extended linearly to all of P defines a multiplication (via the convolution product) on $\mathrm{Hom}(P, K)$. This multiplication is essentially pointwise multiplication of characteristic functions, i.e., intersection of subsets of $\{x, y\}^*$. The multiplicative identity is the language $\{x, y\}^*$, corresponding to the $f \in \mathrm{Hom}(P, K)$ such that $f(w) = 1$ for all monomials w.

8 K-Bialgebras and Running Automata in Parallel

We have seen that a K-algebra A allows us to define automata which take elements of A as input. These automata compute elements of $\mathrm{Hom}(A, K)$. A coalgebra structure on A allows to multiply elements of $\mathrm{Hom}(A, K)$. We now discuss the relation between products on $\mathrm{Hom}(A, K)$ and automata.

We first treat the case in which A is both a K-algebra and a K-coalgebra, without assuming that A is a K-bialgebra. Let $C = (M, A, s_C, \triangleright_C, \alpha_C)$ and $D = (N, A, s_D, \triangleright_D, \alpha_D)$ be automata. Applying the convolution product to ρ_C and ρ_D yields

$$\rho_C * \rho_D(a) = \mu_K \circ (\sum_i \rho_C(a_{(1)} \triangleright s_C) \otimes \rho_D(a_{(2)} \triangleright s_D)).$$

Consider the following comultiplications defined on the monomials of P:

$$\Delta_1(w) = w \otimes w$$

$$\Delta_2(w) = \sum_{w_1 w_2 = w} w_1 \otimes w_2$$

extended linearly to P. Also consider the comultiplication defined as

$$\Delta_3(x) = 1 \otimes x + x \otimes 1$$

$$\Delta_3(y) = 1 \otimes y + y \otimes 1$$

and extended to all of P as a bialgebra. We have seen that Δ_1 yields the intersection of two languages. A simple calculation shows that Δ_2 yields concatenation of languages and Δ_3 the shuffle product (see [4] and also [15], Proposition 5.1.4). In other words, the convolution product determines a formula with comultiplication as a parameter. Different choices of comultiplication yield different products of languages. When the languages are given by automata, we can use this formula to obtain a succinct expression for the product of the two languages.

Of course, it would be even better if we could get an *automaton* accepting the product of the two languages. For a K-bialgebra, there is an easy way to construct such an automaton. It relies on a construction from the theory of bialgebras.

We emphasize that a bialgebra structure is not necessary for an automaton accepting $\rho_C * \rho_D$ to exist. Consider Δ_2 and Δ_3. They agree on x and y, which generate P as an algebra, so at most one of them can be an algebra map; Δ_3 is an algebra map by definition. Therefore Δ_2 is not part of a bialgebra, and so we cannot use the construction to get an automaton accepting the concatenation of two languages. Such an automaton exists, of course, but it is not given by this construction.

Suppose now that B is a K-bialgebra. The first step is to define an action of B on $M \otimes N$ from actions of B on M and N.

Definition 8.1. *Let B be a K-bialgebra which acts on K-semimodules M and N. Then B acts on $M \otimes N$ according to the formula*

$$b \triangleright_{M \otimes N} (m \otimes n) = \sum_i b_{(1)} \triangleright_M m \otimes b_{(2)} \triangleright_N n.$$

See Chapter 1 of [15] for a proof that this is an action.

Definition 8.2. *Let $C = (M, B, s_C, \triangleright_C, \alpha_C)$ and $D = (N, B, s_D, \triangleright_D, \alpha_D)$ be automata over a K-bialgebra B. The* tensor product *of C and D, denoted $C \otimes D$, is the automaton $(M \otimes N, B, s_C \otimes s_D, \triangleright_{M \otimes N}, \alpha_C \otimes \alpha_D)$.*

Remark 8.1. Since $K \otimes K \cong K$, $\alpha_M \otimes \alpha_N : M \otimes N \to K$.

Theorem 8.1. *Let $C = (M, B, s_C, \triangleright_C, \alpha_C)$ and $D = (N, B, s_D, \triangleright_D, \alpha_D)$ be left automata over a K-bialgebra B. Then $\rho_{C \otimes D} = \rho_C * \rho_D$.*

Proof.

$$\rho_{C\otimes D}(b) = \alpha_{C\otimes D}(b \triangleright_{C\otimes D} (s_C \otimes s_D))$$
$$= \alpha_{C\otimes D}(\sum_i b_{(1)} \triangleright_C s_C \otimes b_{(2)} \triangleright_D s_D)$$
$$\sum_i \alpha_C(b_{(1)} \triangleright_C s_C)\alpha_D(b_{(2)} \triangleright_D s_D)$$
$$= \rho_C * \rho_D(b).$$

In the classical case, this corresponds to "running two automata in parallel".

9 Algebras to Automata

Finally, we show how to define an automaton on an arbitrary K-algebra A.

Lemma 9.1. *Let A be a K-algebra which acts on a K-semimodule M from the left. For $a \in A, x \in M$, let $h_a(x) = ax$. Then $Hom(M, K)$ is a right A-semimodule via*

$$f \triangleleft a = f \circ h_a(x)$$

for $f \in Hom(M, K), a \in A$.

Proof. This is another standard algebraic fact. To see that the map $\triangleleft : \mathrm{Hom}(M, K) \otimes A \to \mathrm{Hom}(M, K)$ is K-linear, just verify that the associated map $\mathrm{Hom}(M, K) \times A \to \mathrm{Hom}(M, K)$ is K-bilinear. The function $\triangleleft$ defines a right action on $\mathrm{Hom}(M, K)$ because

$$f \triangleleft ab(x) = f(abx) = (f \circ h_a) \circ h_b(x) = (f \triangleleft a) \triangleleft b(x).$$

for $a, b \in A, x \in M$.

Given a K-algebra A, multiplication defines a left K-semimodule action of A on itself, and hence a right K-semimodule action of A on $\mathrm{Hom}(A, K)$.

Theorem 9.1. *Let K be a commutative semiring and A a K-algebra. Let $f \in Hom(A, K)$. Then $(Hom(A, K), A, f, \triangleleft, \alpha)$ is an automaton, where*

$$f \triangleleft a = f \circ h_a(x),$$
$$\alpha(f) = f(1).$$

Proof. This follows from Lemma 9.1 and the verification that $\alpha : \mathrm{Hom}(A, K) \to K$ is K-linear, which is straightforward.

Example 9.1. Let P be as in Example 5.1. As shown in Example 7.1, we can view elements of $\mathrm{Hom}(P, K)$ as formal languages over $\{x, y\}$. For $f \in \mathrm{Hom}(P, K), \alpha(f)$ is 1 if and only if the associated language contains the empty word, and 0 otherwise. For each $f \in \mathrm{Hom}(P, K)$, $f \circ h_x(w) = f(xw)$. That is,

$$f \circ h_x(w) = 1 \leftrightarrow f(xw) = 1.$$

In other words, $f \triangleleft x$ is the Brzozowski derivative of f with respect to the letter x.

10 Conclusion and Future Work

We have shown that automata and formal languages can be expressed naturally using the language of bialgebras, and that the two theories have several analogous structures and constructions. The next step is to use this correspondence to solve problems of interest to computer science. We give two examples that we hope to develop.

Proofs and Proof Complexity

Automaton morphisms are interesting for their proof-theoretic properties. Given any two equivalent (classical) finite nondeterministic automata C and D (without ϵ-transitions), there exists a sequence of automata and automaton morphisms in the indicated directions witnessing the equivalence:

$$C \leftarrow \text{accessible dfa} \rightarrow \text{minimal dfa} \leftarrow \text{accessible dfa} \rightarrow D.$$

The automaton morphisms can be encoded as matrices over the two-element idempotent semiring. Here "accessible dfa" refers to the dfa obtained by the subset construction, with the inaccessible states removed. This means that automaton morphisms can be used as the sole "rule of inference" in a complete proof system for (classical) finite automaton equivalence [18]. The above proofs are exponential in size in the worst case, because of determinization. However, given two equivalent finite automata, there may be other, shorter proofs. A trivial example is the fact that two isomorphic automata are related by an automaton morphism; there is no need to determinize. We hope to use the theory of (bi)algebras to understand the question: "What makes it difficult to prove two equivalent automata equivalent?" in this proof system. We would also like to investigate the relation between final automata and coinduction [16] and the automaton in Example 9.1.

Other Types of Automata

We plan to investigate other types of automata using this framework - for example, stochastic automata, quantum automata [3] or automata on guarded strings [11]. The hope is to provide a general framework in which generalizes related concepts in the theory of automata and "automaton-like" structures, and also to adapt the above proof system to these cases.

Acknowledgements

The author would like to thank Anil Nerode for many inspiring discussions and the anonymous reviewers for their helpful comments and suggestions. This work was supported by NSF grant CCF-0635028.

References

1. Adàmek, J., Trnkovà, V.: Automata and algebras in categories. Kluwer Academic Publishers, Dordrecht (1990)
2. Buchholz, P.: Bisimulation relations for weighted automata. Theoretical Computer Science 393, 109–123 (2008)
3. Crutchfield, J.P., Moore, C.: Quantum automata and quantum grammars. Theoretical Computer Science 237, 275–306 (2000)
4. Duchamp, G., Flouret, M., Laugerotte, È., Luque, J.-G.: Direct and dual laws for automata with multiplicities. Theoretical Computer Science 267, 105–120 (2001)
5. Duchamp, G., Tollu, C.: Sweedler's duals and Schützenberger's calculus. Arxiv Preprint. arXiv:0712.0125v2
6. Fitting, M.: Bisimulations and boolean vectors. Advances in Modal Logic 4, 97–125 (2003)
7. Golan, J.S.: Semirings and their applications. Kluwer Academic Publishers, Dordrecht (1999)
8. Grossman, R.L., Larson, R.G.: Bialgebras and realizations. In: Bergen, J., Catoiu, S., Chin, W. (eds.) Hopf Algebras, pp. 157–166. Marcel Dekker, Inc., New York (2004)
9. Grossman, R.L., Larson, R.G.: The realization of input-output maps using bialgebras. Forum Mathematicum 4, 109–121 (1992)
10. Katsov, Y.: Tensor products and injective envelopes of semimodules over additively regular semirings. Algebra Colloquium 4(2), 121–131 (1997)
11. Kozen, D.: Automata on guarded strings and applications. Matématica Contemporânea 24, 117–139 (2003)
12. Kozen, D.: A completeness theorem for Kleene algebras and the algebra of regular events. Infor. and Comput. 110(2), 366–390 (1994)
13. Lang, S.: Algebra: revised, 3rd edn. Springer, Heidelberg (2002)
14. Litvinov, G.L., Masloc, V.P., Shpiz, G.B.: Tensor products of idempotent semimodules. An algebraic approach. Mathematical Notes 65(4) (1999)
15. Majid, S.: Foundations of quantum group theory. Cambridge University Press, Cambridge (1995)
16. Rutten, J.J.M.M.: Universal coalgebra: a theory of systems. Theoretical Computer Science 249, 3–80 (2000)
17. Street, R.: Quantum groups: a path to current algebra. Cambridge University Press, Cambridge (2007)
18. Worthington, J.: Automatic proof generation in Kleene algebra. In: Berghammer, R., Möller, B., Struth, G. (eds.) RelMiCS/AKA 2008. LNCS, vol. 4988, pp. 382–396. Springer, Heidelberg (2008)

Author Index