Hiroakira Ono
Makoto Kanazawa
Ruy de Queiroz (Eds.)

Logic, Language, Information and Computation

LNAI 5514

16th International Workshop, WoLLIC 2009
Tokyo, Japan, June 2009
Proceedings

Springer

Lecture Notes in Artificial Intelligence 5514

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

Subseries of Lecture Notes in Computer Science

FoLLI Publications on Logic, Language and Information

Hiroakira Ono Makoto Kanazawa
Ruy de Queiroz (Eds.)

Logic, Language, Information and Computation

16th International Workshop, WoLLIC 2009
Tokyo, Japan, June 21-24, 2009
Proceedings

Springer

Volume Editors

Hiroakira Ono
Japan Advanced Institute of Science and Technology
1–1 Asahidai, Nomi, Ishikawa 923–1292, Japan
E-mail: ono@jaist.ac.jp

Makoto Kanazawa
National Institute of Informatics
2–1–2 Hitotsubashi, Chiyoda-ku, Tokyo 101–8430, Japan
E-mail: kanazawa@nii.ac.jp

Ruy de Queiroz
Universidade Federal de Pernambuco, Centro de Informática
Recife, PE, Brazil
E-mail: ruy@cin.ufpe.br

Library of Congress Control Number: Applied for

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

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-02260-X Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02260-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12693528 06/3180 5 4 3 2 1 0

Preface

The Workshop on Logic, Language, Information and Computation (WoLLIC) has met every year since 1994 with the aim of fostering interdisciplinary research in pure and applied logic. The idea is to have a forum which is large enough in the number of possible interactions between logic and the sciences related to information and computation, and yet is small enough to allow for concrete and useful interaction among participants.

This volume contains the texts of the 25 contributed papers selected out of 57 submissions for presentation at WoLLIC 2009. It also includes six papers written or coauthored by the invited speakers. Between them they give a representative sample of some of the most active areas of research on the frontiers between computation, logic, and linguistics.

We are grateful to all the people who made this meeting possible and are responsible for its success: the members of the Program Committee and the external reviewers, the invited speakers, the contributors, and the people who were involved in organizing the workshop.

We would also like to express our gratitude to the following organizations for supporting WoLLIC 2009: the Association for Symbolic Logic (ASL), the Interest Group in Pure and Applied Logics (IGPL), the European Association for Logic, Language and Information (FoLLI), the European Association for Theoretical Computer Science (EATCS), the Sociedade Brasileira de Computação (SBC), and the Sociedade Brasileira de Lógica (SBL).

The reviewing for the workshop and the preparation of the proceedings were greatly aided by the free EasyChair conference management system, for which we are extremely grateful to its main developer, Andrei Voronkov.

April 2009

Hiroakira Ono
Makoto Kanazawa
Ruy de Queiroz

Organization

WoLLIC 2009 Program Committee

Toshiyasu Arai	Kobe University, Japan
Matthias Baaz	Vienna University of Technology, Austria
Alexandru Baltag	Oxford University, UK
Josep Maria Font	University of Barcelona, Spain
Silvio Ghilardi	University of Milan, Italy
Katsumi Inoue	National Institute of Informatics, Japan
Marcus Kracht	University of Bielefeld, Germany
Hiroakira Ono	Japan Advanced Institute of Science and Technology, Chair
Masanao Ozawa	Nagoya University, Japan
John Slaney	Australian National University, Australia
Mark Steedman	University of Edinburgh, UK
Hans Tompits	Vienna University of Technology, Austria

Additional Reviewers

Stefano Aguzzoli	Mauro Ferrari	Francesco Paoli
Patrick Allo	Camillo Fiorentini	Joerg Puehrer
Mutsunori Banbara	Amélie Gheerbrant	Bryan Renne
Alexander Bochman	Angel Gil	Eike Ritter
Richard Booth	Rajeev Gore	Chiaki Sakama
Felix Bou	Ramon Jansana	Marko Samer
Gerhard Brewka	Yusuke Kubota	Sonja Smets
Anna Bucalo	Thomas Lukasiewicz	Sylvie Thiebaux
Martin Caminada	Pierluigi Minari	Satoshi Tojo
Giovanna D'Agostino	Franco Montagna	Mirek Truszczynski
Anuj Dawar	Sergei Odintsov	Stefan Woltran
Paul Dekker	Johannes Oetsch	Akihiro Yamamoto
Robert Demolombe	Mario Ornaghi	Satoshi Yamane
Uwe Egly	Valeria De Paiva	Jonathan Zvesper

WoLLIC 2009 Organizing Committee

Makoto Kanazawa	National Institute of Informatics, Japan (Co-chair)
Anjolina de Oliveira	Federal University of Pernambuco, Brazil
Ruy de Queiroz	Federal University of Pernambuco, Brazil (Co-chair)
Ken Satoh	National Institute of Informatics, Japan

WoLLIC Steering Committee

Samson Abramsky, Johan van Benthem, Joe Halpern, Wilfrid Hodges, Daniel Leivant, Angus Macintyre, Grigori Mints, Ruy de Queiroz

Table of Contents

Tutorials and Invited Talks

Contributed Papers

A Characterisation of Definable NP Search Problems in Peano Arithmetic

Arnold Beckmann*

Department of Computer Science
Swansea University
Swansea SA2 8PP, UK
a.beckmann@swansea.ac.uk

Abstract. The complexity class of $\prec$-*bounded local search problems with goals* is introduced for well-orderings $\prec$, and is used to give a characterisation of definable NP search problems in Peano Arithmetic.

1 Introduction

A *search problem* in general is just a binary relation R. The search task is to find, given x as input, some y satisfying $R(x, y)$. Search problems play a special role in complexity theory. Usually, they are ignored, that is, studied through corresponding decision problems. Often this leads to satisfying results, for example when the reduction is given by a natural self-reduction which produces a polynomially equivalent decision problem. However, there are situations where this approach is unsatisfying as the decision problem is not computationally equivalent. This is particularly important if we are concerned with *total search problems*, that is, search problems which satisfy $(\forall x)(\exists y)R(x, y)$.

Total NP *search problems* are those where R is polynomial time computable and polynomially bounded — the latter means that $R(x, y)$ always implies that the length of y is polynomially bounded in the length of x. Johnson, Papadimitriou and Yannakakis [JPY88] have initiated the study of total NP search problems, and in particular identified several natural subclasses of total NP search problems depending on the mathematical principle needed to proof their totality.

The totality of NP search problems, or in general the totality of definable (multi-)functions, is also an important theme in the study of logical theories, like fragments of arithmetic, in particular Bounded Arithmetic. Bounded Arithmetic has been introduced by Buss [Bus86] as first-order theories of arithmetic with a strong connection to computational complexity. These theories can be given as restrictions of Peano Arithmetic in a suitable language. A main goal in the study of Bounded Arithmetic is to give natural descriptions of the class of total search problems / (multi-)functions whose totality can be shown within some theory of Bounded Arithmetic [Bus86, Kra93, BK94, Pol99]. Recently, some advances have

* Supported in part by EPSRC grant EP/D03809X/1, and by a grant from the John Templeton Foundation.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 1–12, 2009.

been made in providing characterisations for missing pairs of level of definability and theories of Bounded Arithmetic [KST07, ST07, Pud06, BB08]. In particular, characterisations have been obtained using a machinery which originated from the proof-theoretic study of Peano Arithmetic, using so called *proof notations for continuous cut-elimination* [AB08, BB09].

At this point, it natural to ask whether it is also possible to obtain natural descriptions of those total NP search problems whose totality can be proven in stronger theories than Bounded Arithmetic.[1] The present paper is a first contribution to this programme by studying the definable NP search problems of Peano Arithmetic, and characterising them in terms of a kind of generalised local search problems which we denote formalised α-*bounded local search problems* for $\alpha < \epsilon_0$. Of course, it is not surprising that α ranges over ordinal notations for the ordinal ϵ_0, as ϵ_0 is the well-known proof-theoretic ordinal for Peano Arithmetic, first implicitly established by Gentzen [Gen36] in his consistency proof for Peano Arithmetic.

The next section will briefly introduce Peano Arithmetic in a way suitable for our proof-theoretic investigations. Section 3 defines the search problem classes of α-bounded local search. This is followed in Section 4 by the definition of an ordinal notation system of order-type ϵ_0. Section 5 briefly reviews necessary definitions and results on notations and cut-reduction for Peano Arithmetic from [AB08]. This is followed by the section defining the search problems which come from proofs in Peano Arithmetic, and stating our main result concerning the characterisation of definable NP search problems in terms of α-bounded local search for $\alpha < \epsilon_0$.

2 Peano Arithmetic

Our definition of Peano Arithmetic is based on Bounded Arithmetic, as we want to make use of the machinery developed in [AB08]. Also, we want to obtain in later sections notation systems which have polynomial time computable ingredients, which in particular means that closed terms in the language must evaluate in polynomial time. Thus, allowing symbols for stronger functions than polynomial time computable ones is problematic.

Our proof-theoretic investigations are very much independent of the exact choice of the language. Therefore, we will be very liberal and allow symbols for all polynomial time computable functions. We introduce Bounded Arithmetic very briefly, and in a slightly nonstandard way similar to [AB08]. The reader interested in the general theory of Bounded Arithmetic is kindly referred to the literature [Bus86].

For $a \in \mathbb{N}$ let $|a|$ denote the length of the binary representation of a. We will use $|\cdot|$ also as a symbol for a unary function in the next definition. This will never lead to confusion.

[1] This question has also been formulated in a draft of a book by Pavel Pudlák. The author would like to thank Pavel Pudlák for discussing this question during a one week visit of the author at the Mathematical Institute of the Academy of Sciences of the Czech Republic. The author would also like to thank Jan Krajíček, Pavel Pudlák and Neil Thapen for their hospitality during his stay.

Definition 1 (Language of Bounded Arithmetic). The language $\mathcal{L}_{\mathrm{BA}}$ of Bounded Arithmetic *contains as nonlogical symbols* $\{=, \leq\}$ *for the binary relation "equality" and "less than or equal", and a symbol for each polynomial time computable function. In particular,* $\mathcal{L}_{\mathrm{BA}}$ *includes a constant* c_a *for* $a \in \mathbb{N}$ *whose interpretation in the standard model* $\mathbb{N}$ *is* $c_a^{\mathbb{N}} = a$*, and unary function symbols* $|\cdot|$ *whose standard interpretation is given by* $|\cdot|^{\mathbb{N}} \colon a \mapsto |a|$*. We will often write* $\underline{a}$ *instead of* c_a*, and* 0 *for* c_0*.*

Atomic formulas *are of the form* $s = t$ *or* $s \leq t$ *where* s *and* t *are terms.* Literals *are expressions of the form* A *or* $\neg A$ *where* A *is an atomic formula. Formulas are build up from literals by means of* $\wedge$, $\vee$, $(\forall x)$, $(\exists x)$*. The* negation $\neg C$ for a formula C *is defined via de Morgan's laws. Negation extends to sets of formulas in the usual way by applying it to their members individually.*

We will use the following abbreviations.

Definition 2. *The expression* $A \to B$ *denotes* $\neg A \vee B$*. Bounded quantifiers are introduced as follows:* $(\forall x \leq t)A$ *denotes* $(\forall x)(x \leq t \to A)$, $(\exists x \leq t)A$ *denotes* $(\exists x)(x \leq t \wedge A)$*, where* x *may not occur in* t*.*

Definition 3 (Bounded Formulas). *The set of* bounded $\mathcal{L}_{\mathrm{BA}}$-formulas *is the set of* $\mathcal{L}_{\mathrm{BA}}$*-formulas consisting of literals and being closed under* $\wedge$, $\vee$, $(\forall x \leq t)$, $(\exists x \leq t)$*.*

Definition 4. *The set* $\mathrm{s}\Sigma_1^{\mathrm{b}}$ *consists of all literals and all formulas of the form* $(\exists x \leq s)A(x)$ *where* A *is a literal.* A*,* s *and* t *may depend on other variables not mentioned here.*

Definition 5. *As axioms we allow all disjunctions of literals, i.e., all disjunctions* A *of literals such that* A *is true in* $\mathbb{N}$ *under any assignment. Let us denote this set of axioms by* $\overline{\mathrm{BASIC}}$*.*

The set $\overline{\mathrm{BASIC}}$ is not recursive. Although this is nonstandard for usual formulation of Bounded Arithmetic [Bus86], it is quite normal for the type of proof theoretic invistigations we are after, i.e. using notations for infinitary derivations. It comes from the fact that the complexity of the set of axioms (measured by its arithmetic complexity) of a formal system does not influence the complexity of cut-elimination (measured by the ordinal height of infinitary derivation trees) in the corresponding infinitary propositional derivations.

Definition 6. *Let* $\mathrm{Ind}(A, z, t)$ *denote the expression*

$$A_z(0) \wedge (\forall z \leq t)(A \to A_z(z+1)) \to A_z(t) \ .$$

Definition 7. *Let* S_2^1 *denote the theory consisting (of universal closures) of formulas in* $\overline{\mathrm{BASIC}}$ *and (of universal closures) of formulas of the form* $\mathrm{Ind}(A, z, |t|)$ *with* $A \in \mathrm{s}\Sigma_1^{\mathrm{b}}$*,* z *a variable and* t *an* $\mathcal{L}_{\mathrm{BA}}$*-term.*

Let PA *denote the theory consisting (of universal closures) of formulas in* $\overline{\mathrm{BASIC}}$ *and (of universal closures) of formulas of the form* $\mathrm{Ind}(A, z, t)$ *with* A *an* $\mathcal{L}_{\mathrm{BA}}$ *formula (not necessarily bounded),* z *a variable and* t *an* $\mathcal{L}_{\mathrm{BA}}$*-term.*

Definition 8. *Let Σ_1^{b} be the set of formulas φ such that there exist $\psi \in \mathrm{s}\Sigma_1^{\mathrm{b}}$ with $\mathrm{S}_2^1 \vdash \varphi \leftrightarrow \psi$.*

Let Δ_1^{b} be the set of formulas φ such that there exist formulas σ, π with $\sigma, \neg\pi \in \mathrm{s}\Sigma_1^{\mathrm{b}}$ and $\mathrm{S}_2^1 \vdash (\varphi \leftrightarrow \sigma) \wedge (\varphi \leftrightarrow \pi)$.

3 Bounded Local Search with Goals

A binary relation $R \subseteq \mathbb{N} \times \mathbb{N}$ is called *polynomially bounded* iff there is a polynomial p such that $(x, y) \in R$ implies $|y| \leq p(|x|)$. R is called *total* if for all x there exists a y with $(x, y) \in R$.

Definition 9 (Total and Definable NP Search Problems). *Let $R \subseteq \mathbb{N} \times \mathbb{N}$ be a polynomially bounded, total relation which is polynomial time computable. The* (total) NP search problem *associated with R is this: Given input $x \in \mathbb{N}$, return a $y \in \mathbb{N}$ such that $(x, y) \in R$. R is called* definable in a theory T, *if there exists a $\mathrm{s}\Sigma_1^{\mathrm{b}}$-formula $(\exists y)\varphi(x, y)$ (the bound to y is implicit in φ) with all free variables shown, such that $(x, y) \in R$ iff $\mathbb{N} \vDash \varphi(x, y)$, and such that $T \vdash (\forall x)(\exists y)\varphi(x, y)$.*

A binary relation $\prec$ on $\mathbb{N} \times \mathbb{N}$ is a *polynomial time computable well-ordering,* if it satisfies the conditions that it is polynomial time computable as a binary relation, that it is a total order, and that it is well-founded, i.e. does not contain infinite descending sequences.

We now define the class of $\prec$-bounded local search problems with goals. It will be defined similar to *polynomial local search (PLS) problems* as introduced by Johnson, Papadimitriou, and Yannakakis [JPY88], and in particular Π_k^{p}-PLS with Π_ℓ^{p}-goals from [BB08, BB09]. The main difference will be that the set of possible solutions is not required to be polynomially bounded. We discuss below immediate consequences of this, after we have given the next definition.

Definition 10 ($\prec$-BLS Problems with Goals). *Let $\prec$ be a polynomial time computable well-ordering. A* $\prec$-bounded local search ($\prec$-bls) problem with goal *is a tuple $L = (S, G, d, N, c, i)$ consisting of, for a given input x, a set $S(x)$ of* possible solutions, *a* goal set *$G(x)$ with a polynomial bound d, a* neighbourhood function *$N(x, s)$ mapping a solution s to another solution, a function $c(x, s)$ computing the* cost of a solution *s according to the well-ordering $\prec$, and a function $i(x)$ computing an* initial solution, *such that the functions N, c and i and the predicates F and G are polynomial time computable, and the following six conditions are satisfied:*

$$\prec \text{ is a total order.} \tag{3.1}$$

$$(\forall x, s)(s \in G(x) \rightarrow |s| \leq d(|x|)) \tag{3.2}$$

$$(\forall x)(i(x) \in S(x)) \tag{3.3}$$

$$(\forall x, s)(s \in S(x) \rightarrow N(x, s) \in S(x)) \tag{3.4}$$

$$(\forall x, s)(N(x, s) = s \vee c(x, N(x, s)) \prec c(x, s)) \tag{3.5}$$

$$(\forall x, s)(s \in G(x) \leftrightarrow (N(x, s) = s \wedge s \in S(x))) \tag{3.6}$$

The search task is, for a given input x, to find some s with $s \in G(x)$.

If the well-ordering is understood from the context, we often refer to it by its ordertype given as an ordinal, and e.g. speak of α-bounded local search problems with goals.

We have introduced F and G as sets. They will usually be given via a corresponding relation, e.g. "$s \in S(a)$" in terms of $S(a, s)$.

The following fact is obvious.

Fact 11. *Any $\prec$-bls problem with goal defines a total* NP *search problem in the sense of Definition 9.*

The next observation is almost obvious, and uses the fact that the set of possible solutions is not necessarily polynomially bounded.

Observation 12. *Any total* NP *search problem can be defined by some $<$-bls problem with goal, where $<$ is the natural ordering on* $\mathbb{N}$.

Proof. The proof is based on a simple padding idea. As the set of possible solutions is not required to be polynomially bounded, we first increase the size of a possible solution to reach a possible solution which is exponentially bigger that the polynomially bound of our goal set. At this point it is feasible to directly search for a solution in the goal set.

To be more precise, let R be a total binary relation, which is polynomially bounded using some polynomial d. We define a $<$-bls problem with goal $L = (S, G, d, N, c, i)$ which defines the NP search problem associated with R in the sense of Definition 10: let $b := 2^{d(|x|)}$ (which implies $d(|x|) < |b|$) and define $G(x) := \{y\colon |y| \leq d(|x|) \text{ and } R(x, y)\}$, $S(x) := \{\langle x, b, n\rangle : n \in \mathbb{N}\} \cup G(x)$, $N(x, \langle x, b, n\rangle) := \langle x, b, n^2 + 2\rangle$ if $|n| < b$, $N(x, \langle x, b, n\rangle) := y$ if $|n| \geq b$ and y smallest with $R(x, y)$ (observe that in this case $y < b \leq |n| \leq |\langle x, b, n\rangle|$, thus it is feasible to search for y,) $N(x, s) := s$ otherwise, $i(x) := \langle x, b, 0\rangle$ and $c(x, \langle x, b, n\rangle) := 1 + (b \dot{-} |n|)$, $c(x, s) := 0$ otherwise. □

The previous fact and observation show that the general formulation of $\prec$-bls problems with goals cannot be used to make any meaningful assertions about total NP search problems. That is, they do not lead to a meaningful combinatorial description of a kind of local search problem, which expresses the totality of the overall search problem in some natural way. If we study the previous proof we can see why this is the case: in order to obtain that the neighbourhood function as defined in the previous proof is a well-defined function (that is, is total) we have to know for the step $N(x, \langle x, b, n\rangle) := y$ if $|n| \geq b$ and y smallest with $R(x, y)$, that a y with $R(x, y)$ exists, which means that at this point we already have to invest that the R defines a total NP search problem. And for the *proof* of existence it does not help that n is very big.

One way to ensure that the description of a $\prec$-bounded local search problem stays in some sense "purely combinatorial", is to require that all its conditions can be formalised in some weak theory suitable for formalising combinatorics. We follow this line of thought in the following definition by taking as such theory S^1_2.

Definition 13 (Formalised $\prec$-BLS Problems). *A $\prec$-bls problem with goal in the sense of Definition 10 is* formalised *provided the predicates S, G and $\prec$ are given by Δ_1^{b}-formulas, and the defining conditions (3.1)–(3.6) are provable in S_2^1.*

4 Ordinal Notations for ϵ_0

Let $<$ denote the 'real' semantic concept of ordinal orderings. Recall the Cantor normal form for ordinals; i.e., every ordinal $\alpha > 0$ can be written uniquely in the form

$$\alpha = \omega^{\alpha_1} + \omega^{\alpha_2} + \omega^{\alpha_3} + \cdots + \omega^{\alpha_k},$$

where $k \geq 1$ and $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \cdots \geq \alpha_k$. This is the basis for the well-known representation of ordinals less than ϵ_0: namely, write an ordinal $\alpha < \epsilon_0$ as a term in Cantor normal form, recursively writing the exponents of ω in the same form. We repeat the definition of compact representations for ordinals less than ϵ_0 as given in [BBP03].

Definition 14. *We simultaneously and inductively define a set of expressions, called* normal compact forms for ordinals less than ϵ_0*, and a binary relation $\prec_{\epsilon_0}$ on normal compact forms, as follows, where "$=$" denotes identity on strings:*

1. *If $\alpha_1, \ldots, \alpha_k$ are normal compact forms, and $n_1, \ldots, n_k \in \mathbb{N} \setminus \{0\}$, then the expression $\omega^{\alpha_1} \cdot n_1 + \cdots + \omega^{\alpha_k} \cdot n_k$ is a normal compact form. For $k = 0$ this is the empty word which we denote by 0.*
2. *$\omega^{\alpha_1} \cdot n_1 + \cdots + \omega^{\alpha_k} \cdot n_k \prec_{\epsilon_0} \omega^{\beta_1} \cdot m_1 + \cdots + \omega^{\beta_\ell} \cdot m_\ell$ holds if and only if there is some i with $0 \leq i \leq \min\{k, \ell\}$, such that $\alpha_j = \beta_j$ and $n_j = m_j$ for $j = 1, \ldots, i$, and one of the following cases is satisfied:*
   (a) either $i = k < \ell$; or
   (b) $i < \min\{k, \ell\}$ and $\alpha_{i+1} \prec_{\epsilon_0} \beta_{i+1}$; or
   (c) $i < \min\{k, \ell\}$, $\alpha_{i+1} = \beta_{i+1}$ and $n_{i+1} < m_{i+1}$.

We also write $\alpha \prec_{\epsilon_0} \epsilon_0$ to indicate that α is a normal compact form.

It can be shown (cf. [BBP03]) that S_2^1 can formalise the notion of normal compact forms by using standard sequence coding methods to define the Gödel number of a normal compact form. We assume that some efficient method of sequence coding is used for Gödel numbers so that the length of the Gödel number of a basic form α is proportional to the number of symbols in α.

In this way, the set of normal compact forms and the relation $\prec_{\epsilon_0}$ can be seen to be polynomial time computable based on their inductive definitions, and that the bounded arithmetic theory S_2^1 can Δ_1^{b}-define the syntactic concepts of normal compact forms and the relation $\prec_{\epsilon_0}$, see [BBP03] for more details.

It is also easy to see that the operations $\alpha, \beta \mapsto \alpha + \beta$ of addition and $\alpha \mapsto 3^\alpha$ of exponentiation to base 3 on ordinals can be represented on normal compact forms by polynomial time computable functions. Also observe that the embedding of $\mathbb{N}$ into normal compact forms, given by $n \mapsto \omega^0 \cdot n$, is polynomial time computable.

Finally, we show that S_2^1 can prove that $\prec_{\epsilon_0}$ is a total ordering on normal compact forms, satisfying transitivity and trichotomy.

Theorem 15. *Let α be a normal compact form. The α-bls problems with goals are definable* NP *search problems in* PA.

Proof. Let $L = (S, G, N, c, i)$ be an α-bls problem with goal. Let x be given. The set $A := \{c(x,s)\colon s \in S(x)\}$ is a non-empty subset of $\{\beta\colon \beta \prec_{\epsilon_0} \alpha\}$ by (3.3) and (3.4), and can be expressed by a Σ_1 formula. PA proves transfinite induction up to $\alpha \prec_{\epsilon_0} \epsilon_0$ for Σ_1 properties [Poh09]. Thus, arguing in PA, we can choose some $c \in A$ which is $\prec_{\epsilon_0}$-minimal. Pick $s \in S(x)$ with $c(x,s) = c$, and let $s' := N(x,s)$. Then $s' \in S(x)$ by (3.4). By construction $c(x,s') \not\prec_{\epsilon_0} c(x,s)$, hence (3.5) shows $s' = N(x,s) = s$. Hence, (3.6) shows $s \in G(x)$. □

5 Notation System for Peano Arithmetic

In [AB08], a general framework has been developed which is suitable to characterise definable search problems / (multi-)functions in Bounded Arithmetic. This framework is based on *notations for propositional proofs.* In principle, the same framework can also be used to characterise the definable NP search problems of Peano Arithmetic. The main difference between the notation system for Bounded Arithmetic and that for Peano Arithmetic is that heights of propositional proofs can become infinite in the case of Peano Arithmetic, and therefore have to be bounded by ordinals.

Due to the lack of space, we will only briefly introduce proof notations, and mainly state the differences between those for Bounded Arithmetic and those needed for Peano Arithmetic. The reader interested in more details is kindly referred to [AB08].

A proof notation system is a set (of proof terms) which is equipped with some functions, most prominently a function computing the last inference $\mathrm{tp}(h)$ of a proof named by some notation h, and a function that, given a notation h and a natural number i computes some notation $h[i]$ for the i'th subproof of the derivation named by h. So, a proof notation completely determines an explicit propositional derivation tree; the tree can be reconstructed by exploring it from its root and determining the inference at each node of the tree.

The cut-reduction operator can be defined on the names for derivation trees. Using continuous cut-elimination, these transformations will be particularly simple on the names; note that, using names, for derivations it makes sense to ask about the complexity of getting the i'th subderivation, or about the size of the name, even if it denotes an infinite object. It has been shown [AB08] that the cut-reduction operator on proof notations can be understood as a polynomial time operation. Continuous normalisation for infinitary propositional proofs has been invented by Mints [Min78, KMS75]. The approach in [AB08] is build on Buchholz' technical very smooth approach to notation systems for continuous cut-elimination [Buc91, Buc97].

In [AB08], a notation system $\mathcal{H}_{\mathrm{BA}}$ has been defined which denotes propositional derivations obtained by translating [Tai68, PW85] Bounded Arithmetic proofs. Applying the machinery of notations for continuous cut-elimination, a notation system $\mathcal{CH}_{\mathrm{BA}}$ of cut-elimination for $\mathcal{H}_{\mathrm{BA}}$ has been obtained which has

the property that its implicit descriptions, most notably the functions mentioned above, will be polynomial time computable.

To obtain a similar notation system for Peano Arithmetic we can proceed as follows. Let $\mathcal{F}_{\mathrm{PA}}$ be the set of closed formulas in $\mathcal{L}_{\mathrm{BA}}$. We define the outermost connective function $\mathrm{tp}(f)$ for $f \in \mathcal{F}_{\mathrm{PA}}$ to be $\top$ or $\bot$ for true or false literals, respectively, $\bigwedge$ for universally quantified formulas and conjunctions, and $\bigvee$ for existentially quantified formulas and disjunctions. The sub-formula function $f[n]$ for $f \in \mathcal{F}_{\mathrm{PA}}$ and $n \in \mathbb{N}$ is defined in the obvious way, where for finite conjunctions and disjunctions the last conjunct or disjunct is treated as if it were repeated infinitely often. The rank $\mathrm{rk}(A)$ of a formula A in $\mathcal{F}_{\mathrm{PA}}$ is defined in the usual way measuring its depth: $\mathrm{rk}(A) := 0$ for atomic formulas A, for $A = B \wedge C$ or $A = B \vee C$ let $\mathrm{rk}(A) := 1 + \max\{\mathrm{rk}(B), \mathrm{rk}(C)\}$. If $A = (\forall x)B$ or $A = (\exists x)B$, let $\mathrm{rk}(A) := 1 + \mathrm{rk}(B)$.

As closed terms are evaluated to numbers when translating PA-proofs into propositional ones, notations have to be considered modulo the natural intentional equivalence relation $\approx_{\mathbb{N}}$ which identifies terms with the same value. As our definition of $\mathcal{L}_{\mathrm{BA}}$ only contains function symbols for polynomial time computable functions, $\approx_{\mathbb{N}}$ will be polynomial time decidable if the depth of expressions is restricted, and the number of function symbols representing polynomial time functions is also restricted to a finite subset.

Let PA^{∞} denote the propositional proof system over $\mathcal{F}_{\mathrm{PA}}$. The last inference of a derivation in PA^{∞} can be of the form (Ax_A) for $A \in \mathcal{F}_{\mathrm{PA}}$ with $\mathrm{tp}(A) = \top$ indicating an axiom, $(\bigwedge_C)$ for $C \in \mathcal{F}_{\mathrm{PA}}$ with $\mathrm{tp}(C) = \bigwedge$ indication an application of a $\bigwedge$-inference with main formula C, $(\bigvee_C^i)$ for $C \in \mathcal{F}_{\mathrm{PA}}$ with $\mathrm{tp}(C) = \bigvee$ and $i \in \mathbb{N}$ indicating an application of a $\bigvee$-inference with main formula C and side formula $C[i]$, (Cut_C) for $C \in \mathcal{F}_{\mathrm{PA}}$ with $\mathrm{tp}(C) \in \{\top, \bigwedge\}$ indicating an application of a cut inference, and the void repetition inference (Rep) which neither introduces nor discharges a formula.

The *finitary proof system* $\mathrm{PA}^{\star}$ is some particularly nice formal proof system for first order logic, which includes also some special rules for induction. It is mainly given by the same inference symbols as $\mathrm{BA}^{\star}$ in [AB08].

Finally, let $\mathcal{H}_{\mathrm{PA}}$ be the set of closed $\mathrm{PA}^{\star}$-derivations. For each $h \in \mathcal{H}_{\mathrm{PA}}$ we define the denoted last inference $\mathrm{tp}(h)$ and subderivations $h[j]$ following the obvious translation into propositional logic, were induction up to 2^i is proved by a balanced tree of cuts of height i. The height $\mathrm{o}(h)$ is defined according to the above description of a tree of balanced cuts; the increase of the height caused by one application of induction can be bounded by ω. The cut-rank $\mathrm{crk}(h)$ of a derivation $h \in \mathcal{H}_{\mathrm{PA}}$ is defined as usual by strictly bounding the ranks of all cut-formulas. We write $h \vdash_{\approx_{\mathbb{N}}} \Gamma$ to indicate that Γ is a superset (modulo $\approx_{\mathbb{N}}$) of the end-sequent of the propositional derivation denoted by h.

As for $\mathcal{H}_{\mathrm{BA}}$ [AB08], we can now add notations for cut-elimination to obtain $\mathcal{CH}_{\mathrm{PA}}$. In particular, we add a symbol E which represents the reduction of cuts by one level, and which has the following properties: If $h \vdash_{\approx_{\mathbb{N}}} \Gamma$, then $\mathsf{E}h \vdash_{\approx_{\mathbb{N}}} \Gamma$, $\mathrm{crk}(\mathsf{E}h) \leq \mathrm{crk}(h) \dot{-} 1$ and $\mathrm{o}(\mathsf{E}h) = 3^{\mathrm{o}(h)}$.

As in the case of $\mathcal{H}_{\mathrm{BA}}$ it can be seen that all functions involved in $\mathcal{H}_{\mathrm{PA}}$ and $\mathcal{CH}_{\mathrm{PA}}$ are polynomial-time computable.

Theorem 16. *Assume* $\mathrm{PA} \vdash \varphi$ *with* $\mathrm{FV}(\varphi) \subseteq \{x\}$. *Then, there is some* $\mathrm{PA}^\star$*-derivation* h *such that* $\mathrm{FV}(h) \subseteq \{x\}$, $h \vdash_{\approx_{\mathbb{N}}} \varphi$, *and* $\mathrm{o}(h(\underline{a}/x)) \prec_{\epsilon_0} \omega \cdot 2$.

6 Definable NP Search Problems in Peano Arithmetic

We start by describing the idea for computing witnesses using proof trees. Assume we have a PA-proof of a formula $(\exists y)\varphi(y)$ in $\mathrm{s}\Sigma^{\mathrm{b}}_1$ and we want to compute an n such that $\varphi(n)$ is true — in case we are interested in definable search problems, such a situation is obtained from a proof of $(\forall x)(\exists y)\varphi(x, y)$ by inverting the universal quantifier to some $a \in \mathbb{N}$. Assume further, that we have applied cut-elimination to obtain a PA^∞ derivation d_0 of $(\exists y)\varphi(y)$ with $\mathrm{crk}(d_0) = 0$. Then we can define a path through d_0, represented by sub-derivations $d_1, d_2, d_3 \ldots$, such that d_j is an immediate sub-derivation of d_{j+1}, and the end-sequent of d_j is of the form $(\exists y)\varphi(y), \Gamma_j$ where all formulas $A \in \Gamma_j$ are false and either atomic or instances of sub-formulas of $(\exists y)\varphi(y)$. Such a path must be finite as the height of d_j is strictly decreasing. Say it ends with some d_ℓ. In this situation we must have that last inference of d_ℓ is $\bigvee^k_{(\exists y)\varphi(y)}$ and that $\varphi(k)$ is true. Hence we found our witness.

Before we capture this idea in Definition 18 we will define a function on proof notations that computes the next step in the path described above.

Definition 17. *Let* $\mathcal{CH}^k_{\mathrm{PA}}$ *denote the set of notations* h *in* $\mathcal{CH}_{\mathrm{PA}}$ *which satisfy that the index of any function symbols occurring in* h *is bounded by* k, *and that the depths of any formula or term occurring in* h *is also bounded by* k *(the depth of constants is counted as* 0*.)*

We define a function $\mathrm{red}\colon \mathcal{CH}^k_{\mathrm{PA}} \cup \{0\} \to \mathcal{CH}^k_{\mathrm{PA}} \cup \{0\}$ *by* $h \mapsto \mathrm{red}(h)$ *with*

$$\mathrm{red}(h) := \begin{cases} 0 & \text{if } h = 0 \text{ or } \mathrm{tp}(h) = \mathrm{Ax}_A \text{ or} \\ & \mathrm{tp}(h) = \bigvee^d_{(\exists y)\varphi(\underline{a},y)} \text{ with } \varphi(\underline{a}, \underline{d}) \text{ true,} \\ h[1] & \text{if } \mathrm{tp}(h) = \mathrm{Cut}_C \text{ with } C \text{ true,} \\ h[1] & \text{if } \mathrm{tp}(h) = \bigwedge_{A_0 \wedge A_1} \text{ with } A_0 \wedge A_1 \text{ of the form} \\ & \varphi(\underline{a}, \underline{d}) \text{ for some } d \text{ and } A_0 \text{ true,} \\ h[0] & \text{otherwise.} \end{cases}$$

It is clear from the introduction of proof notations for PA that red is polynomial time computable.

Definition 18. *We define a* parameterised α-bounded local search problem *by* $k \in \mathbb{N}$, $\alpha \prec_{\epsilon_0} \epsilon_0$, *a* $\mathrm{PA}^\star$*-derivation* h *which defines an* initial value function

$$h(\cdot)\colon \mathbb{N} \to \mathcal{CH}_{\mathrm{PA}},\ a \mapsto h(a) := \underbrace{\mathsf{E} \cdots \mathsf{E}}_{\mathrm{crk}(h)\times} h(\underline{a}/x)\ ,$$

and a formula $(\exists y)\varphi(x,y) \in \mathrm{s}\Sigma^{\mathrm{b}}_1$*, such that* S^1_2 *proves, for* $a \in \mathbb{N}$*, that* $h(a) \vdash_{\approx_{\mathbb{N}}} (\exists y)\varphi(\underline{a}, y)$*,* $\mathrm{crk}(h(a)) = 0$*, and* $\mathrm{o}(h(a)) \prec_{\epsilon_0} \alpha$*. We denote such a parametrisation by* $P = \langle k, \alpha, h, (\exists y)\varphi(x,y)\rangle$*.*

This parametrisation defines an α*-bounded local search problem with goal* $L = (S, G, d, N, c, i)$ *in the following way: Let* $t(x)$ *be the bound to* y *which is implicit in* $(\exists y)\varphi(x,y)$*. An instance is given by some* $a \in \mathbb{N}$*. The goal set is defined as* $G(a) := \{y\colon \varphi(\underline{a}, y)\}$*; the set of possible solutions as*

$$S(a) := G(a) \cup \{\langle t(a), h_0, \ldots, h_\ell\rangle : h_0 = h(a),\ h_\ell \neq 0 \text{ and } (\forall i{<}\ell)\, h_{i+1} = \mathrm{red}(h_i)\}\ ;$$

the neighbourhood function is defined as

$$N(a, \langle t(a), h_0, \ldots, h_\ell\rangle) := \begin{cases} \langle t(a), h_0, \ldots, h_\ell, \mathrm{red}(h_\ell)\rangle & \text{if } \mathrm{red}(h_\ell) \neq 0 \\ d & \text{if } \mathrm{red}(h_\ell) = 0 \text{ and } \mathrm{tp}(h_\ell) = \bigvee^d_{(\exists y)\varphi(x,y)} \end{cases}$$

$$N(a, d) := d \qquad \text{for } d \leq t(a)\ ;$$

the initial value function *is given by* $i(a) := \langle t(a), h(a)\rangle$*; and the* cost function *is defined as* $c(a, \langle t(a), h_0, \ldots, h_\ell\rangle) := \mathrm{o}(h_\ell)$*, and* $c(a, d) := 0$ *for* $d < D_a$*.*

Proposition 19. *The local search problem* $L = (S, G, d, N, c, i)$ *parameterised by* $P = \langle k, \alpha, h, (\exists y)\varphi(x,y)\rangle$ *from Definition 18 provides an* α*-bls problem with goal according to Definition 10 which solves* φ*.*

Theorem 20. *The definable* NP *search problems in* PA *can be characterised by formalised* α*-bls problems with goals for* $\alpha \prec_{\epsilon_0} \epsilon_0$*.*

Proof. Assume $\mathrm{PA} \vdash (\forall x)(\exists y)\varphi(x,y)$ with $(\exists y)\varphi(x,y) \in \mathrm{s}\Sigma^{\mathrm{b}}_1$. Inverting the $(\forall x)$ quantifier we obtain $\mathrm{PA} \vdash (\exists y)\varphi(x,y)$. By Theorem 16, we obtain some $\mathrm{PA}^\star$-derivation h such that $\mathrm{FV}(h) \subseteq \{x\}$, $h \vdash_{\approx_{\mathbb{N}}} (\exists y)\varphi(x,y)$, and $\mathrm{o}(h(\underline{a}/x)) \prec_{\epsilon_0} \omega \cdot 2$.

Let k be so large that it bounds all indices of function symbols occurring in h, as well as the logical depths of all formulas and terms (where constants have depth 0) occurring in h. Let $\alpha := 3_{\mathrm{crk}(h)}(\omega \cdot 2)$. Then $P = \langle k, \alpha, h, (\exists y)\varphi(x,y)\rangle$ defines a parameterised α-bls problem according to Definition 18, because the following are provable in S^1_2, using $h(a) := \underbrace{\mathsf{E}\cdots\mathsf{E}}_{\mathrm{crk}(h)\times} h(\underline{a}/x)$:

- $h(a) \vdash_{\approx_{\mathbb{N}}} (\exists y)\varphi(\underline{a}, y)$;
- $\mathrm{crk}(h(a)) = \mathrm{crk}(h(\underline{a}/x)) \dot{-} \mathrm{crk}(h) = \mathrm{crk}(h) \dot{-} \mathrm{crk}(h) = 0$;
- $\mathrm{o}(h(a)) = 3_{\mathrm{crk}(h)}(\mathrm{o}(h(\underline{a}/x))) \prec_{\epsilon_0} 3_{\mathrm{crk}(h)}(\omega \cdot 2) = \alpha$.

By Proposition 19, this defines an α-bls problem with goal which solves φ. □

Together with Theorem 15 we obtain the following

Corollary 21. *The definable* NP *search problems in* PA *are exactly characterised by formalised* α*-bls problems with goals for* $\alpha \prec_{\epsilon_0} \epsilon_0$*.*

7 Conclusion

We have characterised the definable NP search problems of Peano Arithmetic in terms of formalised α-bls problems with goals for $\alpha \prec_{\epsilon_0} \epsilon_0$. One immediate question is whether the defining conditions (3.1)–(3.6) can be turned into some independent principle, by rendering all involved polynomial time functions and predicates in a generic way using oracles (cf. [BB08, BB09]).

Further steps in this programme will be to investigate whether it can be extended to stronger theories than PA. The hope would be that for any theory for which a suitable ordinal analysis has been accomplished [Poh09], this can be turned into some feasible notation system which can form the basis of some class of $\prec$-bls problems characterising the definable NP search problems of that theory. A next step here could be to use the description of Γ_0 in [BBP03].

References

[AB08] Aehlig, K., Beckmann, A.: On the computational complexity of cut-reduction. Submitted to APAL (2008)

[BB08] Beckmann, A., Buss, S.R.: Polynomial local search in the polynomial hierarchy and witnessing in fragments of bounded arithmetic. Technical Report CSR15-2008, Department of Computer Science, Swansea University (December 2008)

[BB09] Beckmann, A., Buss, S.R.: Characterising definable search problems in bounded arithmetic via proof notations. Technical report, Department of Computer Science, Swansea University (January 2009)

[BBP03] Beckmann, A., Buss, S.R., Pollett, C.: Ordinal notations and well-orderings in bounded arithmetic. Ann. Pure Appl. Logic 120(1-3), 197–223 (2003)

[BK94] Buss, S.R., Krajíček, J.: An application of Boolean complexity to separation problems in bounded arithmetic. Proc. London Math. Soc. (3) 69(1), 1–21 (1994)

[Buc91] Buchholz, W.: Notation systems for infinitary derivations. Archive for Mathematical Logic 30, 277–296 (1991)

[Buc97] Buchholz, W.: Explaining Gentzen's consistency proof within infinitary proof theory. In: Gottlob, G., Leitsch, A., Mundici, D. (eds.) KGC 1997. LNCS, vol. 1289, pp. 4–17. Springer, Heidelberg (1997)

[Bus86] Buss, S.R.: Bounded arithmetic. Studies in Proof Theory. Lecture Notes, vol. 3. Bibliopolis, Naples (1986)

[Gen36] Gentzen, G.: Die Widerspruchsfreiheit der reinen Zahlentheorie. Math. Ann. 112, 493–565 (1936)

[JPY88] Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comput. System Sci. 37(1), 79–100 (1988); 26th IEEE Conference on Foundations of Computer Science (Portland, OR, 1985)

[KMS75] Kreisel, G., Mints, G.E., Simpson, S.G.: The use of abstract language in elementary metamathematics: Some pedagogic examples. In: Parikh, R. (ed.) Logic Colloquium. Lecture Notes in Mathematics, vol. 453, pp. 38–131. Springer, Heidelberg (1975)

[Kra93] Krajíček, J.: Fragments of bounded arithmetic and bounded query classes. Trans. Amer. Math. Soc. 338(2), 587–598 (1993)

[KST07] Krajíček, J., Skelley, A., Thapen, N.: NP search problems in low fragments of bounded arithmetic. J. Symbolic Logic 72(2), 649–672 (2007)

[Min78] Mints, G.E.: Finite investigations of transfinite derivations. Journal of Soviet Mathematics 10, 548–596 (1978); Translated from: Zap. Nauchn. Semin. LOMI 49 (1975); Cited after Grigori Mints. Selected papers in Proof Theory. Studies in Proof Theory. Bibliopolis (1992)

[Poh09] Pohlers, W.: Proof theory. The first step into impredicativity. Universitext. Springer, Berlin (2009)

[Pol99] Pollett, C.: Structure and definability in general bounded arithmetic theories. Ann. Pure Appl. Logic 100(1-3), 189–245 (1999)

[Pud06] Pudlák, P.: Consistency and games—in search of new combinatorial principles. In: Logic Colloquium 2003. Lect. Notes Log., vol. 24, pp. 244–281. Assoc. Symbol. Logic, La Jolla (2006)

[PW85] Paris, J., Wilkie, A.: Counting problems in bounded arithmetic. In: Dold, A., Eckmann, B. (eds.) Methods in Mathematical Logic (Proceedings Caracas 1983). Lecture Notes in Mathematics, vol. 1130, pp. 317–340. Springer, Heidelberg (1985)

[ST07] Skelley, A., Thapen, N.: The provable total search problems of bounded arithmetic (2007); Typeset manuscript

[Tai68] Tait, W.W.: Normal derivability in classical logic. In: Barwise, J. (ed.) The Syntax and Semantics of Infinitatry Languages. Lecture Notes in Mathematics, vol. 72, pp. 204–236. Springer, Heidelberg (1968)

Algebraic Valuations as Behavioral Logical Matrices*

Carlos Caleiro[1] and Ricardo Gonçalves[1,2]

[1] Instituto de Telecomunicações and Dept. Mathematics, IST, TU-Lisbon, Portugal
[2] CENTRIA, FCT-UNL, Portugal

Abstract. The newly developed behavioral approach to the algebraization of logics extends the applicability of the methods of algebraic logic to a wider range of logical systems, namely encompassing many-sorted languages and non-truth-functionality. However, where a logician adopting the traditional approach to algebraic logic finds in the notion of a logical matrix the most natural semantic companion, a correspondingly suitable tool is still lacking in the behavioral setting. Herein, we analyze this question and set the ground towards adopting an algebraic formulation of valuation semantics as the natural generalization of logical matrices to the behavioral setting, by establishing a few simple but promising results. For illustration, we will use da Costa's paraconsistent logic $\mathcal{C}_1$.

Keywords: algebraic logic, behavioral algebraization, logical matrix, valuation semantics.

1 Introduction

A novel behavioral approach to the algebraization of logics was introduced in [7] with the aim of extending the range of applicability of the traditional tools of algebraic logic. The extended theory is able to provide a meaningful algebraic counterpart also to logics with a many-sorted syntax, or including non-truth-functional connectives, and which are not algebraizable with the usual approach. Intuitively, while the algebraization process is usually centered around the notion of congruence, behavioral algebraization is centered around the weaker notion of behavioral equivalence. Behavioral equivalence has its roots in computer science, namely in the field of algebraic specifications of data-types, where it is often necessary to reason about data which cannot be directly accessed [16]. In such situations, it is perfectly possible that one cannot distinguish between two different values if those values provide exactly the same results for all available ways of observing and experimenting with them. Hence, unsorted equational logic is replaced by many-sorted behavioral equational logic (sometimes called hidden equational logic) based on

* This work was partially supported by FCT and EU FEDER, namely via the project KLog PTDC/MAT/68723/2006 of SQIG-IT. The second author was also supported by FCT under the postdoctoral grant SFRH/BPD/47245/2008.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 13–25, 2009.

the notion of behavioral equivalence, given a set of available experiments. Behavioral reasoning in equational logic has been consistently developed, see for instance [14,19].

As a consequence of the generalization of the process of algebraizing logics to the behavioral setting, however, one finds that the central notion of matrix semantics [15] is no longer adequate. Of course, it is well-known that every structural logic is fully characterized by the class of its matrix models, or even better by the class of its reduced matrix models [20]. In the case of a logic algebraizable according to the traditional methods, one even gets an equational characterization of the algebras underlying these matrix models, a neat characterization of matrix congruences by means of the Leibniz operator, and a way of recovering the corresponding matrix filters by using the defining equations of the algebraization [2,13]. In contrast, in the behavioral approach, this is not such an easy task. First of all, due to the additional freedom provided by the notion of behavioral equivalence, the corresponding behavioral version of the Leibniz operator is in general not a congruence over the whole language of the logic. Moreover, as expected in the case of logics that are not algebraizable under the usual approach (but which may be behaviorally algebraizable), the connection between the logic and its matrix semantics may be weak and uninteresting. A paradigmatic example of this situation can be found in da Costa's system of paraconsistent logic $\mathcal{C}_1$ [8]. In fact, $\mathcal{C}_1$ is well-known not to be algebraizable using traditional means, and additionally all its Lindenbaum matrices are reduced.

Still, the logic $\mathcal{C}_1$ is behaviorally algebraizable, and its resulting behaviorally equivalent algebraic semantics is quite interesting [6]. Namely, with little effort, it allows us to recover the non-truth-functional bivaluation semantics of [9]. Valuations as a general semantic tool were proposed in [10] precisely with the aim of providing a semantic ground for logics that, like $\mathcal{C}_1$, lack a meaningful truth-functional semantics. The key idea is, in the extreme case, to drop the condition that formulas should always be interpreted homomorphically in an algebra over the same signature. Besides lacking a thorough study, namely if contrasted to the myriad of interesting and valuable algebraic theory underlying logical matrices (see [20]), valuation semantics has been criticized for its excessive generality (see, for instance, [12]). Still, everyone would agree that matrix semantics is simply a clever and algebraically well-behaved way of defining a valuation semantics. What we propose in this paper, taking into account the experience with $\mathcal{C}_1$, and as already suggested in [7,5], is to adopt a suitable algebraic version of valuation semantics as the natural generalization of logical matrices to the behavioral setting. Namely, we will drop the requirement that formulas must be interpreted homomorphically, but we will require that there is an algebraic way of specifying these exceptions. This is just preliminary work, in the sense that our proposal will lack a deep body of results, namely as those available about logical matrices in the usual theory of abstract algebraic logic. What we hope, herein, is to set the ground for developing such results about algebraic valuations and the behavioral approach in the near future.

The paper is organized as follows. In Section 2, we fix notation and concepts that will be necessary for the remainder of the paper, most notably behavioral equational reasoning. Section 3 briefly overviews the notion of behavioral algebraizable logic, as well as the behavioral version of the Leibniz operator. Then, in Section 4, we motivate and present the notion of valuation semantics, and some of its properties. Section 5 is dedicated to establishing a few promising results that parallel, for valuations and in the behavioral setting, well known bridging results between logical matrices and traditional algebraization. Finally, in Section 6, we draw some conclusions and point to some topics of future work.

2 Preliminaries

In this work we will focus our attention on a wide class of logics: those whose language can be built from a rich many-sorted signature. A *many-sorted signature* is a pair $\Sigma = \langle S, F\rangle$ where S is a set (of *sorts*) and $F = \{F_{ws}\}_{w\in S^*, s\in S}$ is an indexed family of sets (of *operations*). For simplicity, we write $f : s_1 \dots s_n \to s \in F$ for an element $f \in F_{s_1\dots s_n s}$. As usual, we denote by $T_\Sigma(X) = \{T_{\Sigma,s}(X)\}_{s\in S}$ the S-sorted family of carrier sets of the free Σ-algebra $\mathbf{T_\Sigma(X)}$ with generators taken from a sorted family $X = \{X_s\}_{s\in S}$ of variable sets. We will denote by $x{:}s$ the fact that $x \in X_s$. Often, we will need to write terms over a given finite set of variables $t \in T_\Sigma(x_1 : s_1, \dots, x_n : s_n)$. For simplicity, we will denote such a term by $t(x_1 : s_1, \dots, x_n : s_n)$. Moreover, if T is a set whose elements are all terms of this form, we will write $T(x_1 : s_1, \dots, x_n : s_n)$. A *substitution* over Σ is a S-sorted family of functions $\sigma = \{\sigma_s : X_s \to T_{\Sigma,s}(X)\}_{s\in S}$. As usual, $\sigma(t)$ denotes the term obtained by uniformly applying σ to each variable in t. Given $t(x_1 : s_1, \dots, x_n : s_n)$ and terms $t_1 \in T_{\Sigma,s_1}(X), \dots, t_n \in T_{\Sigma,s_n}(X)$, we will write $t(t_1, \dots, t_n)$ to denote the term $\sigma(t)$ where σ is a substitution such that $\sigma_{s_1}(x_1) = t_1, \dots, \sigma_{s_n}(x_n) = t_n$. Extending everything to sets, given $T(x_1 : s_1, \dots, x_n : s_n)$ and $U \in T_{\Sigma,s_1}(X) \times \dots \times T_{\Sigma,s_n}(X)$, we will use $T[U] = \bigcup_{\langle t_1,\dots,t_n\rangle\in U} T(t_1, \dots, t_n)$. A *derived operation* of type $s_1 \dots s_n \to s$ over Σ is simply a term in $T_{\Sigma,s}(x_1 : s_1, \dots, x_n : s_n)$. For $w \in S^*$, we denote by $Der_{\Sigma,ws}$ the set of all derived operations of type $w \to s$ over Σ. A *(full) subsignature* of Σ is a many-sorted signature $\Gamma = \langle S, F'\rangle$ such that, for each $w \in S^*$ and $s \in S$, $F'_{ws} \subseteq Der_{\Sigma,ws}$. Given a many-sorted signature $\Sigma = \langle S, F\rangle$, a *$\Sigma$-algebra* is a pair $\mathbf{A}{=}\langle\{A_s\}_{s\in S}, \_\mathbf{A}\rangle$, where each A_s is a non-empty set, the *carrier of sort s*, and $\_\mathbf{A}$ assigns to each operation $f : s_1 \dots s_n \to s$ a function $\underline{f}_\mathbf{A} : A_{s_1} \times \dots \times A_{s_n} \to A_s$. An assignment over $\mathbf{A}$ is a S-sorted family of functions $h = \{h_s : X_s \to A_s\}_{s\in S}$. As usual, we will often overload h and use it to denote also the unique extension of the assignment to an homomorphism $h : T_\Sigma(X) \to \mathbf{A}$. Given a Σ-algebra $\mathbf{A}$, a term $t(x_1 : s_1, \dots, x_n : s_n)$ and $\langle a_1, \dots, a_n\rangle \in A_{s_1} \times \dots \times A_{s_n}$, then we denote by $t_\mathbf{A}(a_1, \dots, a_n)$ the value $h(t)$ that t takes in $\mathbf{A}$ under an assignment h such that $h(x_1) = a_1, \dots, h(x_n) = a_n$. We denote by $\mathbf{A}|_\Gamma$ the Γ-algebra obtained by forgetting in a given Σ-algebra $\mathbf{A}$ the interpretation of all the operations not in the subsignature Γ. We will use $t \approx u$ to represent an equation between terms $t, u \in T_{\Sigma,s}(X)$ of the same sort

s, in which case we dub it an s-equation. The S-sorted set of all Σ-equations will be written as Eq_Σ. We will denote quasi-equations by $(t_1 \approx u_1)\ \&\ldots\&\ (t_n \approx u_n) \rightarrow (t \approx u)$. A set Θ of equations with variables in $\{x_1 : s_1, \ldots, x_n : s_n\}$ will be dubbed $\Theta(x_1 : s_1, \ldots, x_n : s_n)$. As usual, we say that an assignment h over $\mathbf{A}$ *satisfies* the equation $t \approx u$, in symbols $\mathbf{A}, h \Vdash t \approx u$ if $h(t) = h(u)$. We say that $\mathbf{A}$ satisfies $t \approx u$, in symbols $\mathbf{A} \Vdash t \approx u$, if $\mathbf{A}, h \Vdash t \approx u$ for every assignment h over $\mathbf{A}$. Given a class $\mathbb{K}$ of Σ-algebras, the *equational consequence over Σ associated with* $\mathbb{K}$, $\models_{\mathbb{K}} \subseteq \mathcal{P}(Eq_\Sigma) \times Eq_\Sigma$, is such that $\Theta \models_{\mathbb{K}} t \approx u$ if for every $\mathbf{A} \in \mathbb{K}$ and assignment h over $\mathbf{A}$ we have that $\mathbf{A}, h \Vdash t \approx u$ whenever $\mathbf{A}, h \Vdash \Theta$. Moreover, we say that $\mathbf{A}$ satisfies a quasi-equation $(t_1 \approx u_1)\ \&\ldots\&\ (t_n \approx u_n) \rightarrow (t \approx u)$, denoted by $\mathbf{A} \Vdash (t_1 \approx u_1)\ \&\ldots\&\ (t_n \approx u_n) \rightarrow (t \approx u)$, whenever $\{t_1 \approx u_1, \ldots, t_n \approx u_n\} \models_{\{\mathbf{A}\}} t \approx u$.

As mentioned above, the key ingredient of the behavioral approach to algebraizing logics is to use *behavioral equational logic* in the role usually played by plain *equational logic*. The distinctive feature of behavioral equational logic is the fact that sorts are split in two disjoint sets, of *visible* and *hidden* sorts, and only certain operations of visible sort are allowed as *experiments*. In the visible sorts we can perform simple equational reasoning, but we can only reason indirectly about hidden sorts using *behavioral indistinguishability* under the available experiments. Intuitively, we must evaluate equations involving hidden values using only their visible properties. It may happen that under the available experiments two certain hidden terms always coincide, which makes them behaviorally equivalent, even though they might actually have distinct values. We now put forward the rigorous definitions. A *hidden many-sorted signature* is a tuple $\langle \Sigma, V, \mathcal{E} \rangle$ where $\Sigma = \langle S, F \rangle$ is a many sorted-signature, $V \subseteq S$ is the set of visible sorts, and $\mathcal{E}$ is the set of available *experiments*, that is, a set terms of visible sort of the form $t(x : s, x_1 : s_1, \ldots, x_n : s_n)$ where x is a distinguished variable of hidden sort $s \in H = S \setminus V$.

Definition 1. Consider a hidden many-sorted signature $\langle \Sigma, V, \mathcal{E} \rangle$ and a Σ-algebra $\mathbf{A}$. Given a hidden sort $s \in H$, two values $a, b \in A_s$ *are $\mathcal{E}$-behaviorally equivalent*, in symbols $a \equiv_{\mathcal{E}} b$, if for every experiment $t(x : s, x_1 : s_1, \ldots, x_n : s_n) \in \mathcal{E}$ and every $\langle a_1, \ldots, a_n \rangle \in A_{s_1} \times \ldots \times A_{s_n}$, we have that $t_{\mathbf{A}}(a, a_1, \ldots, a_n) = t_{\mathbf{A}}(b, a_1, \ldots, a_n)$.

Now that we have defined behavioral equivalence, we can talk about behavioral satisfaction of an equation by a Σ-algebra $\mathbf{A}$. We say that an assignment h over $\mathbf{A}$ *$\mathcal{E}$-behaviorally satisfies* an equation $t \approx u$ of hidden sort $s \in H$, in symbols $\mathbf{A}, h \Vdash\!\!\!\vdash^{\mathcal{E}} t \approx u$ if $h(t) \equiv_{\mathcal{E}} h(u)$. Expectedly, equations of visible sort are satisfied as usual, that is, if $t \approx u$ is an equation of sort $s \in V$ then we write $\mathbf{A}, h \Vdash\!\!\!\vdash^{\mathcal{E}} t \approx u$ if $h(t) = h(u)$. These notion can now be smoothly extended. We say that $\mathbf{A}$ $\mathcal{E}$-behaviorally satisfies $t \approx u$, in symbols $\mathbf{A} \Vdash\!\!\!\vdash^{\mathcal{E}} t \approx u$, if $\mathbf{A}, h \Vdash\!\!\!\vdash^{\mathcal{E}} t \approx u$ for every assignment h over $\mathbf{A}$. Given a class $\mathbb{K}$ of Σ-algebras, the *behavioral consequence over Σ associated with $\mathbb{K}$ and $\mathcal{E}$*, $\models\!\!\!\vdash^{\mathcal{E}}_{\mathbb{K}} \subseteq \mathcal{P}(Eq_\Sigma) \times Eq_\Sigma$, is such that $\Theta \models\!\!\!\vdash^{\mathcal{E}}_{\mathbb{K}} t \approx u$ if for every $\mathbf{A} \in \mathbb{K}$ and every assignment h over $\mathbf{A}$ we have that $\mathbf{A}, h \Vdash\!\!\!\vdash^{\mathcal{E}} t \approx u$ whenever $\mathbf{A}, h \Vdash\!\!\!\vdash^{\mathcal{E}} t' \approx u'$ for every $t' \approx u' \in \Theta$. Moreover, we say that $\mathbf{A}$ $\mathcal{E}$-behaviorally satisfies a quasi-equation $(t_1 \approx u_1)\ \&\ldots\&\ (t_n \approx u_n) \rightarrow (t \approx u)$,

denoted by $\mathbf{A} \Vdash^{\mathcal{E}} (t_1 \approx u_1)\ \&\ldots\&\ (t_n \approx u_n) \rightarrow (t \approx u)$, whenever $\{t_1 \approx u_1, \ldots, t_n \approx u_n\} \models^{\mathcal{E}}_{\{\mathbf{A}\}} t \approx u$. We refer the reader to [18] for more details on the subject of behavioral equational reasoning.

3 Behavioral Algebraization

From now on, we will work only with many-sorted signatures $\Sigma = \langle S, F\rangle$ with a distinguished sort ϕ (the syntactic sort of formulas). We assume fixed a S-sorted family X of variables. We define the induced set of *formulas* $L_\Sigma(X)$ to be the carrier set of sort ϕ of the free algebra $\mathbf{T_\Sigma(X)}$ with generators X, that is, $L_\Sigma(X) = T_{\Sigma,\phi}(X)$. We now introduce the class of logics that is the target of our approach.

Definition 2. A *many-sorted logic* is a tuple $\mathcal{L} = \langle \Sigma, \vdash\rangle$ where Σ is a many-sorted signature and $\vdash \subseteq \mathcal{P}(L_\Sigma(X)) \times L_\Sigma(X)$ is a *consequence relation* satisfying, for every $\Phi_1 \cup \Phi_2 \cup \{\varphi\} \subseteq L_\Sigma(X)$: if $\varphi \in \Phi_1$ then $\Phi_1 \vdash \varphi$ (***reflexivity***); if $\Phi_1 \vdash \varphi$ for all $\varphi \in \Phi_2$, and $\Phi_2 \vdash \psi$ then $\Phi_1 \vdash \psi$ (***cut***); if $\Phi_1 \vdash \varphi$ and $\Phi_1 \subseteq \Phi_2$ then $\Phi_2 \vdash \varphi$ (***weakening***). A logic $\mathcal{L}$ is further said to be ***structural*** if whenever $\Phi_1 \vdash \varphi$ then, for every substitution σ, $\sigma[\Phi_1] \vdash \sigma(\varphi)$; and said to be ***finitary*** if whenever $\Phi_1 \vdash \varphi$ then $\Phi_2 \vdash \varphi$ for some finite $\Phi_2 \subseteq \Phi_1$. In this paper, unless otherwise stated, all the logics considered are assumed to be structural.

Note that propositional-like logics appear as a particular case of many-sorted logics. They can be obtained by taking ϕ to be the only sort of the signature, that is, considering a signature $\Sigma = \langle S, F\rangle$ such that $S = \{\phi\}$. In the sequel, we will use $\vdash_{\mathcal{L}}$ instead of just $\vdash$ to refer to the consequence relation of a given logic $\mathcal{L} = \langle \Sigma, \vdash\rangle$. Moreover, as usual, if $\Phi_1, \Phi_2 \subseteq L_\Sigma(X)$, we will write $\Phi_1 \vdash_{\mathcal{L}} \Phi_2$ whenever $\Phi_1 \vdash_{\mathcal{L}} \varphi$ for all $\varphi \in \Phi_2$. We say that $\varphi, \psi \in L_\Sigma(X)$ are *interderivable* in $\mathcal{L}$, which is denoted by $\varphi \dashv\vdash_{\mathcal{L}} \psi$, if $\varphi \vdash_{\mathcal{L}} \psi$ and $\psi \vdash_{\mathcal{L}} \varphi$. Analogously, we say that Φ_1 and Φ_2 are *interderivable* in $\mathcal{L}$, which is denoted by $\Phi_1 \dashv\vdash_{\mathcal{L}} \Phi_2$, if $\Phi_1 \vdash_{\mathcal{L}} \Phi_2$ and $\Phi_2 \vdash_{\mathcal{L}} \Phi_1$. The *theorems* of $\mathcal{L}$ are the formulas φ such that $\emptyset \vdash_{\mathcal{L}} \varphi$. A *theory* of $\mathcal{L}$ is a set of formulas Φ such that if $\Phi \vdash_{\mathcal{L}} \varphi$ then $\varphi \in \Phi$. As usual, $\Phi^{\vdash_{\mathcal{L}}}$ denotes the least theory of $\mathcal{L}$ that contains Φ.

In the present setting, given the signature $\Sigma = \langle S, F\rangle$ of a logic, the corresponding free many-sorted algebra will have a set of terms of each sort, but only those terms of sort ϕ will correspond to formulas of the logic. Therefore, in the logic itself, one can only observe the behavior of terms of other sorts by their indirect influence on the formulas where they appear. The behavioral approach to the algebraization of logics is built over the idea of taking this situation a step further. Namely, we will hide all the sorts of Σ, including ϕ, and introduce a new unique visible sort for observing the behavior of formulas. Experiments must be carefully chosen among the well-behaved connectives of the logic, determined by a given subsignature Γ of Σ, thus possibly allowing the remaining connectives to behave in a non-congruently. This can be achieved by considering behavioral equational logic over an extended many-sorted signature. We define the extended signature $\Sigma^o = \langle S^o, F^o\rangle$ such that $S^o = S \uplus \{v\}$, where

v is the newly introduced sort of *observations* of formulas. The indexed set of operations $F^o = \{F^o{}_{ws}\}_{w\in(S^o)^*,s\in S^o}$ is such that $F^o{}_{ws} = F_{ws}$ if $w \in S^*$ and $s \in S$, $F^o_{\phi v} = \{o\}$, and $F^o_{ws} = \emptyset$ otherwise. Intuitively, we are just extending the signature with a new sort v for the observations that we can perform on formulas using the observation operation o. Finally, the extended hidden signature is $\langle \Sigma^o, \{v\}, \mathcal{E}_\Gamma\rangle$ where $\mathcal{E}_\Gamma = \{o(t(x:s, x_1:s_1, \ldots, x_m:s_m)) : t \in T_{\Gamma,\phi}(X)\}$. Henceforth, we will use Γ instead of $\mathcal{E}_\Gamma$ to qualify the corresponding notions of behavioral reasoning. Before we recall the notion of a behaviorally algebraizable logic, we need a further concept. Let $\Theta(x:\phi)$ be a set of ϕ-equations. Θ is said to be Γ-*compatible* with a class $\mathbb{K}$ of Σ^o algebras if, given any variable $y:\phi$, it is the case that $x \approx y, \Theta(x) \models^{\Gamma}_{\mathbb{K}} \Theta(y)$.

Definition 3. A many-sorted logic $\mathcal{L} = \langle \Sigma, \vdash\rangle$ is *behaviorally algebraizable* if there exists a subsignature Γ of Σ, a class $\mathbb{K}$ of Σ^o-algebras, a set $\Theta(x:\phi)$ of ϕ-equations Γ-compatible with $\mathbb{K}$, and a set $\Delta(x_1:\phi, x_2:\phi) \subseteq T_{\Gamma,\phi}(\{x_1:\phi, x_2:\phi\})$ of formulas such that, for every $\Phi \cup \{\varphi\} \subseteq L_\Sigma(X)$ and for every set $\Psi \cup \{t \approx u\}$ of ϕ-equations, we have:

i) $\Phi \vdash_{\mathcal{L}} \varphi$ iff $\Theta[\Phi] \models^{\Gamma}_{\mathbb{K}} \Theta(\varphi)$;
ii) $\Psi \models^{\Gamma}_{\mathbb{K}} t \approx u$ iff $\Delta[\Psi] \vdash_{\mathcal{L}} \Delta(t,u)$;
iii) $x \dashv\vdash_{\mathcal{L}} \Delta[\Theta(x)]$ and $x_1 \approx x_2 \dashv\models^{\Gamma}_{\mathbb{K}} \Theta[\Delta(x_1,x_2)]$.

The set Θ is called the set of *defining equations*, Δ the set of *equivalence formulas*, and $\mathbb{K}$ a *behaviorally equivalent algebraic semantics* for $\mathcal{L}$. This definition is parameterized by the choice of the subsignature Γ of Σ. We will say that a logic is Γ-behaviorally algebraizable if we want to stress the choice of Γ.

The attentive reader will notice that the statement above follows very closely the usual definition of an algebraizable logic, but with behavioral reasoning replacing the usual equational reasoning. As shown in [7], behavioral algebraization indeed enlarges the scope of the traditional theory of algebraization, but maintains many of its nice properties. Namely, given a Σ-algebra $\mathbf{A}$, there is a very natural way of defining a corresponding Γ-behavioral Leibniz operator $\mathbf{\Omega}^{\mathbf{A}}_{\mathbf{\Gamma}}$ that maps each filter $D \subseteq A_\phi$ to the largest congruence $\mathbf{\Omega}^{\mathbf{A}}_{\mathbf{\Gamma}}(D)$ of $\mathbf{A}|_\Gamma$ that is compatible with D. Note that $\mathbf{\Omega}^{\mathbf{A}}_{\mathbf{\Gamma}}(D)$ is in general not a congruence over $\mathbf{A}$ if Γ is a proper subsignature of Σ. In particular, if we consider the free algebra $\mathbf{T_\Sigma(X)}$, we will write $\mathbf{\Omega_\Gamma}$ instead of $\mathbf{\Omega}^{\mathbf{T_\Sigma(X)}}_{\mathbf{\Gamma}}$. As in the traditional approach, the behavioral Leibniz operator can be used to characterize important classes of logics with respect to their algebraic properties. Namely, a logic $\mathcal{L}$ is Γ-behaviorally algebraizable exactly when, on the theories of $\mathcal{L}$, $\mathbf{\Omega_\Gamma}$ is injective, monotone, and commutes with inverse substitutions [7].

Example 1. As an application of the behavioral approach, let us consider the example of da Costa's paraconsistent logic $\mathcal{C}_1$ of [8]. As shown in detail in [7,16], the logic is behaviorally algebraizable despite the well-known fact that it is not algebraizable according to the usual theory, and that it further lacks a meaningful matrix semantics. We just need to let Γ include all the classical connectives

of the logic, but not its non-congruent paraconsistent negation. Moreover, the resulting behaviorally equivalent algebraic semantics for the logic, $\mathbb{K}_{\mathcal{C}_1}$, is very rich, allowing not only to fully explain the relational semantics provided by the so-called da Costa algebras, but also to recapture its best known semantics based on non-truth-functional bivaluations [9].

In the classical theory of algebraization, the class of models canonically associated with a logic $\mathcal{L}$ is typically not the whole family $\mathrm{Matr}(\mathcal{L})$ of matrix models of $\mathcal{L}$, but rather the subclass $\mathrm{Matr}^*(\mathcal{L})$ of $\mathcal{L}$'s reduced matrices[1]. In general, a matrix for $\mathcal{L}$ can be reduced by simply factoring it with the corresponding Leibniz congruence. This is however a problem in the behavioral case, as typically the behavioral Leibniz congruence is not a congruence over the whole matrix.

4 Algebraic Valuations

Valuation semantics appeared in [10] as an effort to provide a semantic ground to logics that, like $\mathcal{C}_1$, lack a meaningful truth-functional semantics. The underlying idea is to drop the condition that formulas should always be interpreted homomorphically, and instead accept any possible interpretation as a function from the set of formulas of the logic to a set of truth-values equipped with a subset of designated values. Besides lacking a thorough supporting theory, namely if contrasted to the rich theory of logical matrices, valuation semantics has been mostly criticized for its excessive generality, namely as it can be (mis)understood at the light of Suszko's bivalence thesis (see, for instance, [4,12]). What we propose in this paper, is to adopt a suitable algebraic version of valuation semantics as the natural generalization of logical matrices to the behavioral setting. Namely, we drop the requirement that formulas must be interpreted homomorphically, but we still require that a certain regularity is maintained. Given the signature Σ of a many-sorted logic, recall that a Σ-matrix is a pair $\langle \mathbf{A}, D\rangle$ where $\mathbf{A}$ is a Σ-algebra and $D \subseteq A_\phi$ is a set of designated elements. In the matrix semantics setting formulas are interpreted by means of the homomorphic extension of any assignment h over $\mathbf{A}$. The key idea of the valuation semantics is to drop this assumption. There can be operations that are always interpreted homomorphically, but also some that are not. Below, we will consider fixed a many-sorted signature $\Sigma = \langle S, O\rangle$ and a subsignature Γ of Σ.

Definition 4. A *Γ-valuation* is a triple $\vartheta = \langle \mathbf{A}, D, h\rangle$ such that $\langle \mathbf{A}, D\rangle$ is a Γ-matrix, and h is a sorted function $h : T_\Sigma(X) \to A$ such that $h(f(t_1, \ldots, t_n)) = f_{\mathbf{A}}(h(t_1), \ldots, h(t_n))$ for every $f : s_1 \ldots s_n \to s \in \Gamma$ and $t_i \in T_{\Sigma,s_i}(X)$ with $i \in \{1, \ldots, n\}$. A *Γ-valuation semantics* over Σ is a collection $\mathcal{V}$ of Γ-valuations.

[1] We have borrowed here the terminology used, for instance, in [20]. In the modern terminology of algebraic logic [13], the classes $\mathrm{Matr}(\mathcal{L})$ and $\mathrm{Matr}^*(\mathcal{L})$ are instead denoted by $\mathrm{Mod}(\mathcal{L})$ and $\mathrm{Mod}^*(\mathcal{L})$, respectively. We dropped this terminology here, as our main point is precisely that matrices do not provide the most natural semantical approach in the behavioral algebraic setting.

A Γ-valuation is a matrix over the subsignature Γ of Σ together with a function that satisfies the homomorphism condition for every connective in Γ. In other words, $h : \mathbf{T_\Sigma}(\mathbf{X})|_\Gamma \to \mathbf{A}$ must be an homomorphism between Γ-algebras. In this way we are allowing valuations that do not necessarily satisfy the homomorphism condition with respect to connectives outside Γ. Both the notion of matrix semantics and the original notion of valuation semantics are particular cases of this definition. The former can be obtained by taking $\Gamma = \Sigma$ and requiring that, for each relevant Σ-algebra $\mathbf{A}$, every homomorphism $h : \mathbf{T_\Sigma}(\mathbf{X}) \to \mathbf{A}$ is considered. The latter can be obtained by observing that the original valuation semantics assume fixed sets of truth and designated values. So, by letting $\Gamma = \langle S, \emptyset \rangle$ and requiring that all Γ-valuations share the same algebraic reduct, we have an original valuation semantics.

Expectedly, given a Γ-valuation $\vartheta = \langle \mathbf{A}, D, h \rangle$ and a formula $\varphi \in L_\Sigma(X)$, we say that ϑ *satisfies* φ, denoted by $\vartheta \Vdash \varphi$, if $h(\varphi) \in D$. As usual, given $\Phi \subseteq L_\Sigma(X)$, we write $\vartheta \Vdash \Phi$ whenever $\vartheta \Vdash \varphi$ for every $\varphi \in \Phi$. Let $\mathcal{L} = \langle \Sigma, \vdash \rangle$ be a many-sorted logic over Σ, not necessarily structural. The Γ-valuation ϑ is said to be a *Γ-model of $\mathcal{L}$* when it happens that if $\vartheta \Vdash \Phi$ and $\Phi \vdash_\mathcal{L} \varphi$ then $\vartheta \Vdash \varphi$. In this case, D is called a *Γ-deductive filter for $\mathcal{L}$*, or just a *$\mathcal{L}$-Γ-filter*, over $\mathbf{A}$. The set of all Γ-models of $\mathcal{L}$ will be denoted by $\mathrm{Val}_\Gamma(\mathcal{L})$. Given a Γ-valuation semantics $\mathcal{V} = \{\langle \mathbf{A}_i, D_i, h_i \rangle : i \in I\}$ over Σ, we define the consequence relation associated with $\mathcal{V}$, $\vdash_\mathcal{V} \subseteq \mathcal{P}(L_\Sigma(X)) \times L_\Sigma(X)$, by letting $\Phi \vdash_\mathcal{V} \varphi$ if for every Γ-valuation $\vartheta \in \mathcal{V}$ we have that $\vartheta \Vdash \varphi$ whenever $\vartheta \Vdash \Phi$. A Γ-valuation semantics $\mathcal{V}$ is *sound* for $\mathcal{L}$ if $\vdash_\mathcal{L} \subseteq \vdash_\mathcal{V}$. Symmetrically, $\mathcal{V}$ is *adequate* for $\mathcal{L}$ if $\vdash_\mathcal{V} \subseteq \vdash_\mathcal{L}$. The Γ-valuation semantics $\mathcal{V}$ is *complete* for $\mathcal{L}$ if it is both sound and adequate, that is $\vdash_\mathcal{L} = \vdash_\mathcal{V}$.

One can also recast the usual Lindenbaum-Tarski constructions. For each set $\Phi \subseteq L_\Sigma(X)$ of formulas, we can define the Γ-valuation $\vartheta^\Phi_\Gamma = \langle \mathbf{L_\Sigma}(\mathbf{X})|_\Gamma, \Phi, id \rangle$ where $id : L_\Sigma(X) \to L_\Sigma(X)$ is the identity function. The Γ-valuations of the form ϑ^Φ_Γ are dubbed *Lindenbaum Γ-valuations* for Σ. The family $\mathcal{V}_\Gamma(\mathcal{L}) = \{\vartheta^\Phi_\Gamma : \Phi \text{ is a theory of } \mathcal{L}\}$ is called the *Lindenbaum Γ-bundle* of $\mathcal{L}$.

Proposition 1. *For every many-sorted logic $\mathcal{L}$,*

- *$\mathcal{L}$ is complete with respect to its Lindenbaum Γ-bundle;*
- *$\mathcal{L}$ is complete with respect to the class of its Γ-models.*

Proof. Clearly, $\mathcal{V}_\Gamma(\mathcal{L}) \subseteq \mathrm{Val}_\Gamma(\mathcal{L})$. To see that $\mathcal{V}_\Gamma(\mathcal{L})$ is an adequate Γ-valuation semantics for $\mathcal{L}$, just suppose that $\Phi \nvdash_\mathcal{L} \varphi$ for some $\Phi \cup \{\varphi\} \subseteq L_\Sigma(X)$. Then $\vartheta^{\Phi^{\vdash_\mathcal{L}}}_\Gamma \Vdash \Phi$ but $\vartheta^{\Phi^\vdash}_\Gamma \nVdash \varphi$, and hence $\Phi \nvdash_{\mathcal{V}_\Gamma(\mathcal{L})} \varphi$. As a consequence, also $\mathrm{Val}_\Gamma(\mathcal{L})$ is adequate for $\mathcal{L}$. □

As the class of all matrix models of $\mathcal{L}$ in the usual approach, the class $\mathrm{Val}_\Gamma(\mathcal{L})$ is very important since it captures algebraically some of the metalogical properties of $\mathcal{L}$. As we will show below, when a logic is behaviorally algebraizable, we are able to algebraically specify not only the class of algebras associated with the logic, but also the admissible ways that formulas can be interpreted in these algebras, as the valuations are now incorporated in the algebraic models. Note

that it is precisely the extended signature Σ^o that gives the algebraic handle that allows us to specify these. There are, however, other desirable properties that a valuation semantics might enjoy. One semantical property which is very characteristic of the algebraic setting, and which holds for a matrix semantics, is that we can consider all possible assignments over a given algebra. A Γ-valuation semantics $\mathcal{V}$ over Σ is said to be *Laplacian* if whenever $\vartheta = \langle \mathbf{A}, D, h\rangle \in \mathcal{V}$ then for every assignment ρ over $\mathbf{A}$ there exists a Γ-valuation $\vartheta_\rho = \langle \mathbf{A}, D, h'\rangle \in \mathcal{V}$ such that $h'|_X = \rho$. Another typical property of a matrix semantics is *representativity*. A Γ-valuation semantics $\mathcal{V}$ over Σ is is said to be *representative* if $\vartheta = \langle \mathbf{A}, D, h\rangle \in \mathcal{V}$ implies that $\vartheta \circ \sigma = \langle \mathbf{A}, D, h \circ \sigma\rangle \in \mathcal{V}$ for every substitution σ. This last property is well-known to be closely connected with structurality [20].

Theorem 1. *Let $\mathcal{L}$ be a many-sorted logic over signature Σ, not necessarily structural, and Γ a subsignature of Σ. Then, $\mathcal{L}$ is structural iff the class $Val_\Gamma(\mathcal{L})$ is representative.*

Proof. Suppose that $\mathcal{L}$ is structural, let $\vartheta \in \mathrm{Val}_\Gamma(\mathcal{L})$ and take any substitution σ. Assume that $\Phi \vdash_{\mathcal{L}} \varphi$ and $\vartheta \circ \sigma \Vdash \Phi$. Clearly, this is equivalent to having $\vartheta \Vdash \sigma[\Phi]$. But, by structurality, it is also the case that $\sigma[\Phi] \vdash_{\mathcal{L}} \sigma(\varphi)$ and, as $\vartheta \in \mathrm{Val}_\Gamma(\mathcal{L})$, it follows that $\vartheta \Vdash \sigma(\varphi)$. Equivalently, then, $\vartheta \circ \sigma \Vdash \varphi$, and hence $\vartheta \circ \sigma \in \mathrm{Val}_\Gamma(\mathcal{L})$, and $\mathrm{Val}_\Gamma(\mathcal{L})$ is representative.

To prove the converse, given that according to Proposition 1 $\mathcal{L}$ is complete with respect to $\mathrm{Val}_\Gamma(\mathcal{L})$, it suffices to show that the consequence associated with an arbitrary representative valuation semantics $\mathcal{V}$ is necessarily structural. Assume that $\Phi \vdash_{\mathcal{V}} \varphi$ and take any substitution σ. Given $\vartheta \in \mathcal{V}$, if $\vartheta \Vdash \sigma[\Phi]$ then, equivalently, $\vartheta \circ \sigma \Vdash \Phi$. But we know that $\vartheta \circ \sigma \in \mathcal{V}$, and therefore $\vartheta \circ \sigma \Vdash \varphi$, or equivalently, $\vartheta \Vdash \sigma(\varphi)$. Hence, $\sigma[\Phi] \vdash_{\mathcal{V}} \sigma(\varphi)$ and $\vdash_{\mathcal{V}}$ is structural. □

To see that some further important results of the fruitful theory of logical matrices generalize to valuation semantics, we will end this section by presenting an example of such a result, namely an adaptation of Bloom's theorem [3] that characterizes finitary logics in our setting. We start by introducing some necessary extended operations on valuations. Given a set $\Lambda = \{\vartheta_i = \langle \mathbf{A}_i, D_i, h_i\rangle : i \in I\}$ of Γ-valuations the *direct product of* Λ is the Γ-valuation $\Pi_{i\in I}\vartheta_i = \langle \Pi_{i\in I}\mathbf{A}_i, \Pi_{i\in I}D_i, (h_i(\_))_{i\in I}\rangle$. Recall that, given a set I, an *ultrafilter on* I is a set $\mathcal{U}$ consisting of subsets of I such that: 1) $\emptyset \notin \mathcal{U}$; 2) if $A \in \mathcal{U}$ and $A \subseteq B$ then $B \in \mathcal{U}$; 3) if $A, B \in \mathcal{U}$ then $A \cap B \in \mathcal{U}$; 4) if $A \subseteq I$, then either $A \in \mathcal{U}$ or $I \setminus A \in \mathcal{U}$. Note that, axioms 1) and 3), imply that A and $I \setminus A$ cannot both be elements of $\mathcal{U}$. Given $\Lambda = \{\vartheta_i : i \in I\}$ and an ultrafilter $\mathcal{U}$ on I we can define a (sorted) equivalence relation $\sim_{\mathcal{U}}$ on the direct product $\Pi_{i\in I}\vartheta_i$ as follows: $a \sim_{\mathcal{U}} b$ iff $\{i \in I : a_i = b_i\} \in \mathcal{U}$. The *ultraproduct* of the Γ-valuations Λ modulo an ultrafilter $\mathcal{U}$, denoted by $\Pi_{\mathcal{U}}\vartheta_i$, is the quotient of $\Pi_{i\in I}\vartheta_i$ by the equivalence $\sim_{\mathcal{U}}$ (that is indeed a congruence of the underlying Γ-algebra). Concretely, let $\Pi_{\mathcal{U}}\vartheta_i = \langle (\Pi_{i\in I}\mathbf{A}_i)/\mathcal{U}, [\{(a_i)_{i\in I} \in \Pi_{i\in I}A_{i,\phi} : \{i \in I : a_i \in D_i\} \in \mathcal{U}\}]_{\mathcal{U}}, [(h_i(\_))_{i\in I}]_{\mathcal{U}}\rangle$.

Theorem 2. *Let $\mathcal{L}$ be a many-sorted logic over signature Σ, and Γ a subsignature of Σ. Then, $\mathcal{L}$ is finitary iff the class $Val_\Gamma(\mathcal{L})$ is closed under ultraproducts.*

Proof. Suppose first that $\mathcal{L}$ is finitary. Let $\{\vartheta_i : i \in I\} \subseteq \mathrm{Val}_\Gamma(\mathcal{L})$ be a family of Γ-models of $\mathcal{L}$ and $\mathcal{U}$ an ultrafilter on I. We aim to prove that $\Pi_\mathcal{U} v_i \in \mathrm{Val}_\Gamma(\mathcal{L})$. So, suppose that $\Phi \vdash_\mathcal{L} \varphi$ and that $\Pi_\mathcal{U}\vartheta_i \Vdash \Phi$. Since $\mathcal{L}$ is finitary, there must exist $\{\varphi_1, \ldots, \varphi_n\} \subseteq \Phi$ such that $\varphi_1, \ldots, \varphi_n \vdash_\mathcal{L} \varphi$. For each $1 \leq j \leq n$, we have that $\Pi_\mathcal{U}\vartheta_i \Vdash \varphi_j$, and thus $I_j = \{i \in I : \vartheta_i \Vdash \varphi_j\} \in \mathcal{U}$. Since $\mathcal{U}$ is an ultrafilter we have that $I_1 \cap \ldots \cap I_n = \{i \in I : \vartheta_i \Vdash \{\varphi_1, \ldots, \varphi_n\}\} \in \mathcal{U}$. Note also that, since each ϑ_i is a Γ-model of $\mathcal{L}$, $I_1 \cap \ldots \cap I_n \subseteq \{i \in I : \vartheta_i \Vdash \varphi\}$. Since $\mathcal{U}$ is an ultrafilter we have that $\{i \in I : \vartheta_i \Vdash \varphi\} \in \mathcal{U}$ and so $\Pi_\mathcal{U}\vartheta_i \Vdash \varphi$.

Suppose now that $\mathrm{Val}_\Gamma(\mathcal{L})$ is closed under ultraproducts. To prove that $\mathcal{L}$ is finitary let Φ be infinite and assume that $\Phi' \nvdash_\mathcal{L} \psi$, for every finite $\Phi' \subseteq \Phi$. Let I denote the set of all finite subsets of Φ. For each $i \in I$, define $i^* = \{j \in I : i \subseteq j\}$. Using well-known results on ultrafilters [20] we can conclude that there exists an ultrafilter $\mathcal{U}$ over I that contains the family $\{i^* : i \in I\}$. For every $i \in I$, consider the theory $i^{\vdash_\mathcal{L}}$ and let $\vartheta_i = \vartheta_\Gamma^{i^{\vdash_\mathcal{L}}} \in \mathrm{Val}_\Gamma(\mathcal{L})$ be the corresponding Lindenbaum Γ-valuation. Let $\Pi_\mathcal{U}\vartheta_i$ be the ultraproduct of the family by the ultrafilter $\mathcal{U}$. Then, for every $\varphi \in \Phi$ we have that $\{\varphi\}^* \subseteq \{i \in I : \vartheta_i \Vdash \varphi\}$. So, $\{i \in I : \vartheta_i \Vdash \varphi\} \in \mathcal{U}$ for every $\varphi \in \Phi$, and consequently we have that $\Pi_\mathcal{U}\vartheta_i \Vdash \Phi$. But $\{i \in I : \vartheta_i \Vdash \psi\} = \emptyset$ and so $\Pi_\mathcal{U}\vartheta_i \nVdash \psi$. Since $\Pi_\mathcal{U}\vartheta_i \in \mathrm{Val}_\Gamma(\mathcal{L})$ we have that $\Phi \nvdash \varphi$, and we can conclude that $\mathcal{L}$ is finitary. □

The class of models canonically associated with a logic $\mathcal{L}$ is typically not the whole of $\mathrm{Matr}(\mathcal{L})$, but rather the subclass $\mathrm{Matr}^*(\mathcal{L})$ of reduced matrices. In the behavioral setting, we can define an analogous class of reduced Γ-valuation models, by setting $\mathrm{Val}^*_\Gamma(\mathcal{L}) = \{\langle \mathbf{A}, D, h\rangle \in \mathrm{Val}_\Gamma(\mathcal{L}) : \mathbf{\Omega}^\mathbf{A}_\mathbf{\Gamma}(D) \text{ is the identity}\}$. Expectedly, given $\Phi \subseteq L_\Sigma(X)$, we can also define the Γ-valuation $\vartheta^{*\Phi}_\Gamma = \langle (\mathbf{L}_\mathbf{\Sigma}(\mathbf{X})|_\mathbf{\Gamma})/\mathbf{\Omega}_\mathbf{\Gamma}(\Phi), [\Phi]_{\mathbf{\Omega}_\mathbf{\Gamma}(\Phi)}, [\_]_{\mathbf{\Omega}_\mathbf{\Gamma}(\Phi)}\rangle \in \mathrm{Val}^*_\Gamma(\mathcal{L})$. The Γ-valuations of this form are dubbed *reduced Lindenbaum Γ-valuations* for Σ. The family $\mathcal{V}^*_\Gamma(\mathcal{L}) = \{\vartheta^{*\Phi}_\Gamma : \Phi \text{ is a theory of } \mathcal{L}\}$ is called the *reduced Lindenbaum Γ-bundle* of $\mathcal{L}$.

Proposition 2. *For every many-sorted logic $\mathcal{L}$,*

- *$\mathcal{L}$ is complete with respect to its reduced Lindenbaum Γ-bundle;*
- *$\mathcal{L}$ is complete with respect to the class of its reduced Γ-models.*

Proof. Noting that $\mathcal{V}^*_\Gamma(\mathcal{L}) \subseteq \mathrm{Val}^*_\Gamma(\mathcal{L})$, the results follows easily from Proposition 1, once we observe that, for every theory Φ of $\mathcal{L}$ and every $\varphi \in L_\Sigma(X)$, we have that $\vartheta^{*\Phi}_\Gamma \Vdash \varphi$ iff $\vartheta^\Phi_\Gamma \Vdash \varphi$. This equivalence follows easily from the fact that $\mathbf{\Omega}_\mathbf{\Gamma}(\Phi)$ is compatible with Φ. □

5 Some Bridge Results

In this section we give a first step into studying the connection between algebraic valuation semantics and behavioral algebraization. We fix a Γ-behaviorally algebraizable many-sorted logic $\mathcal{L} = \langle \Sigma, \vdash\rangle$ with behaviorally equivalent algebraic semantics $\mathbb{K}$ and defining equations Θ. One important consequence of assuming that $\mathcal{L}$ is Γ-behaviorally algebraizable is that given a Σ^o-algebra $\mathbf{A} \in \mathbb{K}$, and by

setting $D_{\mathbf{A}} = \{a \in A_\phi : \delta^i_{\mathbf{A}}(a) \equiv_\Gamma \epsilon^i_{\mathbf{A}}(a) \text{ for every } i \in I\}$, we can recover a filter such that $\langle \mathbf{A}|_\Sigma, D_{\mathbf{A}}\rangle \in \mathrm{Matr}(\mathcal{L})$. However, as we have discussed, the behavioral Leibniz congruence cannot help us to obtain a corresponding reduced matrix. Still, if we consider instead an assignment h on this matrix and take the corresponding Γ-valuation $\vartheta_{\mathbf{A},h} = \langle \mathbf{A}|_\Gamma, D_{\mathbf{A}}, h\rangle \in \mathrm{Val}(\mathcal{L})$, we can now obtain the reduced Γ-valuation $\vartheta^*_{\mathbf{A},h} \in \mathrm{Val}^*(\mathcal{L})$. We need just quotient $\vartheta_{\mathbf{A},h}$ by the behavioral Leibniz congruence $\boldsymbol{\Omega} = \boldsymbol{\Omega}^{\mathbf{A}|_\Gamma}_\Gamma(D_{\mathbf{A}})$, that is, $\vartheta^*_{\mathbf{A},h} = \langle (\mathbf{A}|_\Gamma)/\boldsymbol{\Omega}, [D_{\mathbf{A}}]_{\boldsymbol{\Omega}}, [\_]_{\boldsymbol{\Omega}} \circ h\rangle$. We define $\mathcal{V}_{\mathbb{K}} = \{\vartheta_{\mathbf{A},h} : \mathbf{A} \in \mathbb{K} \text{ and } h \text{ is an assignment over } \mathbf{A}\}$, and $\mathcal{V}^*_{\mathbb{K}} = \{\vartheta^*_{\mathbf{A},h} : \vartheta_{\mathbf{A},h} \in \mathcal{V}_{\mathbb{K}}\}$. We can now establish a sequence of results relating the behavioral consequence associated with $\mathbb{K}$ and the corresponding valuation semantics. The results generalize well-known bridge results linking matrix semantics with traditional algebraization [13].

Proposition 3. *Given $\Phi \cup \{\psi\} \subseteq L_\Sigma(X)$, we have that*

$$\Phi \vdash_{\mathcal{V}^*_{\mathbb{K}}} \varphi \quad \textit{iff} \quad \Phi \vdash_{\mathcal{V}_{\mathbb{K}}} \varphi \quad \textit{iff} \quad \Theta[\Phi] \models^\Gamma_{\mathbb{K}} \Theta(\varphi).$$

Proof. The property follows straightforwardly from the fact that, given an algebra $\mathbf{A} \in \mathbb{K}$, an assignment h over $\mathbf{A}$ and a formula $\varphi \in L_\Sigma(X)$, we have that $\vartheta^*_{\mathbf{A},h} \Vdash \varphi$ iff $\vartheta_{\mathbf{A},h} \Vdash \varphi$ iff $\mathbf{A}, h \Vdash\!\!\Vdash^\Gamma \Theta(\varphi)$. Proving this equivalence is an easy exercise and its proof will be omitted. □

Corollary 1. *If $\mathcal{L}$ is a Γ-behaviorally algebraizable logic with behaviorally equivalent algebraic semantics $\mathbb{K}$ then,*

- *$\mathcal{L}$ is complete with respect to the class $\mathcal{V}^*_{\mathbb{K}}$;*
- *$\mathcal{L}$ is complete with respect to the class $\mathcal{V}_{\mathbb{K}}$.*

We can further show that $\mathcal{V}^*_{\mathbb{K}}$ enjoys other properties that are typical of matrix semantics.

Proposition 4. *If $\mathcal{L}$ is a Γ-behaviorally algebraizable logic with behaviorally equivalent algebraic semantics $\mathbb{K}$ then $\mathcal{V}^*_{\mathbb{K}}$ is both Laplacian and representative.*

Proof. Suppose that $\vartheta = \langle \mathbf{A}, D, h\rangle \in \mathcal{V}_{\mathbb{K}}$. Then there exists $\mathbf{B} \in \mathbb{K}$ and an assignment h' over $\mathbf{B}$ such that $\vartheta = \vartheta_{\mathbf{B},h'}$.

Consider given an assignment ρ over $\mathbf{A}$. Recall that by construction $\mathbf{A}$ results of a quotient construction from $\mathbf{B}|_\Gamma$. Therefore, it is always possible to choose an assignment h^ρ over $\mathbf{B}$ such that, for every $s \in S$ and every $x \in X_s$, we have that $h^\rho_s(x) \in \rho_s(x)$. We can then conclude that $\vartheta_{\mathbf{B},h^\rho} = \langle \mathbf{A}, D, h^\rho\rangle \in \mathcal{V}^*_{\mathbb{K}}$ and $h^\rho|_X = \rho$. Hence, $\mathcal{V}^*_{\mathbb{K}}$ is Laplacian.

To show that $\mathcal{V}^*_{\mathbb{K}}$ is representative, take any substitution σ. Clearly, $h' \circ \sigma$ is also an assignment over $\mathbf{B}$. Hence, $\vartheta_{\mathbf{B},h'\circ\sigma} = \vartheta_{\mathbf{B},h} \circ \sigma \in \mathcal{V}^*_{\mathbb{K}}$. □

In [7] the authors prove that when a logic is Γ-behaviorally algebraizable we can obtain a behavioral equational specification (even if infinite) of its equivalent algebraic semantics $\mathbb{K}$. Therefore, by construction, we can also obtain a specification of the complete Γ-valuation semantics $\mathcal{V}^*_{\mathbb{K}}$. Moreover, when the logic is finitary and finitely Γ-behaviorally algebraizable the specification mentioned above is also finite.

Example 2. As shown in [7,6], the complete reduced valuation semantics for da Costa's paraconsistent logic $\mathcal{C}_1$ resulting from its behavioral algebraization is the class $\mathcal{V}^*_{\mathbb{K}_{\mathcal{C}_1}}$ of valuations $\langle \mathbf{A}, D, h\rangle$ such that $\mathbf{A}$ is a Boolean algebra, $D = \{\top_{\mathbf{A}}\}$, and $h : L_{\mathcal{C}_1}(X) \to A$ satisfies:

- $h(\boldsymbol{t}) = \top_{\mathbf{A}}$ *and* $h(\boldsymbol{f}) = \bot_{\mathbf{A}}$;
- $h(x \wedge y) = h(x) \sqcap_{\mathbf{A}} h(y)$;
- $h(x \vee y) = h(x) \sqcup_{\mathbf{A}} h(y)$;
- $h(x \supset y) = h(x) \Rightarrow_{\mathbf{A}} h(y)$;
- $-_{\mathbf{A}} h(x) \leq_{\mathbf{A}} h(\neg x)$;
- $h(\neg\neg x) \leq_{\mathbf{A}} h(x)$;
- $h(x^\circ) \sqcap_{\mathbf{A}} h(x) \sqcap_{\mathbf{A}} h(\neg x) = \bot_{\mathbf{A}}$;
- $h(x^\circ) \sqcap_{\mathbf{A}} h(y^\circ) \leq_{\mathbf{A}} h((x \wedge y)^\circ)$;
- $h(x^\circ) \sqcap_{\mathbf{A}} h(y^\circ) \leq_{\mathbf{A}} h((x \vee y)^\circ)$;
- $h(x^\circ) \sqcap_{\mathbf{A}} h(y^\circ) \leq_{\mathbf{A}} h((x \supset y)^\circ)$.

The well-known bivaluation semantics of $\mathcal{C}_1$ from [9] can be easily recovered from these valuations by looking at the underlying irreducible Boolean algebras, as explained in [6].

6 Conclusion

We have introduced an algebraic based notion of valuation semantics that arose naturally in semantical considerations in the novel behavioral approach to the algebraization of logics. We have started to pave the way towards the development of a consistent algebraic theory of valuations, that may mirror in the behavioral setting the role played by matrix semantics in the traditional approach to algebraic logic. The results and characterizations obtained are promising. Of course, this paper raises more questions than it gives answers, but the path seems to be clear despite the difficulties raised by the fact that the model-theory of behavioral equational logic remains relatively unexplored. We refer the reader to [17] for a glimpse of what lays ahead. In any case, the references [2,13,20], among others, will provide the essential guidelines for future work. The experience of da Costa's logic $\mathcal{C}_1$ and the recovery of its non-truth-functional bivaluation semantics from the Boolean-based algebraic valuations obtained by the behavioral algebraization process also suggest that a systematic study of the Birkhoff-like operations over valuations is crucial. A related direction in the development of the behavioral theory of algebraization that will need attention is related with Suszko's reduction. Also alternatives to valuation semantics, as proposed here, should be carefully inspected. In particular, we would like to have characterization results for the classes of valuations resulting from non-deterministic matrices [1], both in the static and dynamic versions, or gaggles [11], as well as bridges to the classes of logics that they characterize. We hope to report on these and related questions in forthcoming papers.

References

1. Avron, A.: Non-deterministic matrices and modular semantics of rules. In: Béziau, J.-Y. (ed.) Logica universalis, pp. 149–167. Birkhäuser, Basel (2005)
2. Blok, W., Pigozzi, D.: Algebraizable logics. Memoirs of the AMS, 77(396) (1989)
3. Bloom, S.L.: Some theorems on structural consequence operations. Studia Logica 34, 1–9 (1975)
4. Caleiro, C., Carnielli, W.A., Coniglio, M.E., Marcos, J.: Two's company: The humbug of many logical values. In: Béziau, J.-Y. (ed.) Logica Universalis, pp. 169–189. Birkhäuser, Basel (2005)
5. Caleiro, C., Gonçalves, R.: An algebraic perspective on valuation semantics (abstract). CLE 30/XV EBL/XIV SLALM, Bulletin of Symbolic Logic (in print) (2008)
6. Caleiro, C., Gonçalves, R.: Behavioral algebraization of da Costa's $\mathcal{C}$-systems. Journal of Applied Non-Classical Logics (in print)
7. Caleiro, C., Gonçalves, R., Martins, M.: Behavioral algebraization of logics. Studia Logica 91(1), 63–111 (2009)
8. da Costa, N.: On the theory of inconsistent formal systems. Notre Dame Journal of Formal Logic 15, 497–510 (1974)
9. da Costa, N., Alves, E.H.: A semantical analysis of the calculi $\mathcal{C}_n$. Notre Dame Journal of Formal Logic 18, 621–630 (1977)
10. da Costa, N., Béziau, J.-Y.: Théorie de la valuation. Logique et Analyse 37(146), 95–117 (1994)
11. Dunn, J.M.: Gaggle theory: an abstraction of Galois connections and residuation, with applications to negation, implication, and various logical operators. In: van Eijck, J. (ed.) JELIA 1990. LNCS, vol. 478, pp. 31–51. Springer, Heidelberg (1991)
12. Font, J.: Taking degrees of truth seriously. Studia Logica (in print)
13. Font, J., Jansana, R., Pigozzi, D.: A survey of abstract algebraic logic. Studia Logica 74(1–2), 13–97 (2003)
14. Goguen, J., Malcolm, G.: A hidden agenda. Theoretical Computer Science 245(1), 55–101 (2000)
15. Łoś, J., Suszko, R.: Remarks on sentential logics. Indagationes Mathematicae 20, 177–183 (1958)
16. Reichel, H.: Behavioural validity of conditional equations in abstract data types. In: Contributions to general algebra 3, Proc. Conf., Vienna 1984, pp. 301–324 (1985)
17. Roşu, G.: A Birkhoff-like axiomatizability result for hidden algebra and coalgebra. Electronic Notes in Theoretical Computer Science 11, 179–196 (1998)
18. Roşu, G.: Hidden Logic. PhD thesis, University of California at San Diego (2000)
19. Roşu, G.: Behavioral abstraction is hiding information. Theoretical Computer Science 327(1-2), 197–221 (2004)
20. Wójcicki, R.: Theory of Logical Calculi. In: Synthese Library. Kluwer Academic Publishers, Dordrecht (1988)

Query Answering in Description Logics: The Knots Approach*

Thomas Eiter[1], Carsten Lutz[2], Magdalena Ortiz[1], and Mantas Šimkus[1]

[1] Institute of Information Systems, Vienna University of Technology
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,ortiz,simkus}@kr.tuwien.ac.at
[2] Fachbereich Mathematik und Informatik, University of Bremen
Bibliothekstraße 1, D-28359 Bremen, Germany
clu@informatik.uni-bremen.de

Abstract. In the recent years, query answering over Description Logic (DL) knowledge bases has been receiving increasing attention, and various methods and techniques have been presented for this problem. In this paper, we consider knots, which are an instance of the mosaic technique from Modal Logic. When annotated with suitable query information, knots are a flexible tool for query answering that allows for solving the problem in a simple and intuitive way. The knot approach yields optimal complexity bounds, as we illustrate on the DLs $\mathcal{ALCH}$ and $\mathcal{ALCHI}$, and can be easily extended to accommodate other constructs.

1 Introduction

The recent use of Description Logics (DLs) in a widening range of fields like the Semantic Web, data and information integration and ontology-based data access, has led to the study of new reasoning problems. In particular, accessing semantically enhanced data schemata expressed by means of DL ontologies via (extensions of) the popular *conjunctive queries (CQs)* has become an active area of research, cf. [3,9,8,11,18]. CQs are in general not expressible in the language of most DLs (at least not succinctly), and suitable methods for answering CQs are not always apparent.

The *mosaic technique* is a well-known method in Modal Logics [17,2], which has also been applied for reasoning in DLs [14,24]. Mosaics are small 'blocks' for building models, and possibly infinite models are represented by finite sets of these blocks. Local consistency conditions on mosaics and global coherency condition on sets of mosaics ensure correct model representation. As only finitely many mosaics must be considered and the global and local conditions are effectively verifiable, model existence can be decided by finding a suitable set of mosaics.

Here we discuss an instance of the mosaic technique called *knots* [6]. Knots are small tree-shaped mosaics easy to employ for solving the DL knowledge base

* This work has been partially supported by the Austrian Science Fund (FWF) grants P20840 and P20841, the EU Project Ontorule (FP7 231875), and the Mexican National Council for Science and Technology (CONACYT) grant 187697.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 26–36, 2009.

satisfiability problem. An attractive feature of this method is that it can be gracefully extended to CQ answering: we mark each knot with a *set of (sub)queries* that cannot be mapped locally into the model part the knot describes, and global conditions on sets of marked knots ensure that a full countermodel for the query can be constructed from them.

In this paper, we illustrate the approach on the DLs $\mathcal{ALCH}$ and $\mathcal{ALCHI}$, and obtain a worst-case optimal algorithm for CQ answering in both logics in a transparent way. For illustration of the core technique and to keep matters simple, we focus on a restricted case considering knowledge bases with a very simple data component (i.e., ABox), and give a general outline of how the technique extends to the general setting. As we will see, the marking of the knots is simple and intuitive, and flexible enough to easily extend to other DLs with different constructs. Furthermore, it allows for elegant refinements that yield optimal bounds even in the presence of very subtle sources of complexity.

The rest of the paper is organized as follows. After introducing the DLs we consider and their query answering problem in Section 2, we describe the knot-based algorithm in Section 3. This is done in three steps. First, we describe the generic knot-marking algorithm, and then we show how it can be used in the two considered DLs to obtain optimal complexity. The extensions to the case of unrestricted knowledge bases and to other DLs are briefly described next. Finally, related worked is addressed in Section 4.

2 Preliminaries

We introduce the DLs $\mathcal{ALCH}$ and $\mathcal{ALCHI}$ considered in this paper and discuss the basics of conjunctive query answering. We start with defining knowledge bases. Let $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{I}$ be countably infinite sets of *concept names*, *role names*, and *individual names*. A *role* is either a role name $r \in \mathbf{R}$ or an expression r^- (called the *inverse role of* r). *Concepts* are defined inductively: (i) all concept names $A \in \mathbf{C}$ are concepts, and (ii) if C, D are concepts and r is a role, then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\forall r.C$ and $\exists r.C$ are concepts. As usual, for an inverse role $r = s^-$, we write r^- to denote s.

An $\mathcal{ALCHI}$ *knowledge base (KB)* $\mathcal{K}$ is a finite set of statements of the following form: (i) *concept inclusions (CIs)* $C \sqsubseteq D$ with C and D concepts; (ii) *role inclusions (RIs)* $r \sqsubseteq s$ with r and s roles; (iii) *concept assertions* $A(a)$, with a an individual name and A a concept name; and (iv) *role assertions* $r(a, b)$, with a, b individual names and r a role. If $\mathcal{K}$ does not contain inverse roles, then it is an $\mathcal{ALCH}$ *knowledge base*. By $\mathbf{C}_\mathcal{K}$ and $\mathbf{R}_\mathcal{K}$, we denote the sets of all concept and role names, respectively, that occur in $\mathcal{K}$.

The semantics is given via first-order interpretations. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$ (the *domain*) and a *valuation function* $\cdot^\mathcal{I}$ that maps each concept name $A \in \mathbf{C}$ to subset $A^\mathcal{I}$ of $\Delta^\mathcal{I}$, each role name $r \in \mathbf{R}$ to a subset $r^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$, and each individual name a to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. The function $\cdot^\mathcal{I}$ is extended to all concepts and roles as follows:

$$
\begin{aligned}
(p^-)^{\mathcal{I}} &= \{(y,x) \mid (x,y) \in r^{\mathcal{I}}\},\\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}},\\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},\\
(\exists r.C)^{\mathcal{I}} &= \{x \mid \exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\},\\
(\forall r.C)^{\mathcal{I}} &= \{x \mid \forall y.(x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}.
\end{aligned}
$$

An interpretation $\mathcal{I}$ *satisfies* a CI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, an RI $r \sqsubseteq s$ if $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$, a concept assertion $A(a)$ if $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and a role assertion $r(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. Moreover, $\mathcal{I}$ is a *model of a KB* $\mathcal{K}$ *(in symbols,* $\mathcal{I} \models \mathcal{K}$*)* if it satisfies all inclusions and assertions in $\mathcal{K}$. A KB is *consistent* if it has a model.

We now recall conjunctive queries. To keep the presentation simple, we restrict ourselves to Boolean conjunctive queries, i.e., queries without answer variables. Technically, the general case is no more difficult than this restricted one and admits the same techniques and similar algorithms. We also consider queries with variables only, and constants can be simulated in the usual way. Let $\mathbf{V}$ be a countably infinite set of *variables*. A *(Boolean) conjunctive query (CQ)* is a finite set of atoms of the form $A(x)$ or $r(x,y)$, where A is a concept name, r is a role, and $x, y \in \mathbf{V}$. The variables occurring in the atoms of q are denoted $\mathbf{V}(q)$. When we work with $\mathcal{ALCH}$ KBs, we assume that CQs contain only role names, but no inverse roles.

A *match for* q *in an interpretation* $\mathcal{I}$ is a mapping $\pi : \mathbf{V}(q) \rightarrow \Delta^{\mathcal{I}}$ such that (i) $\pi(x) \in A^{\mathcal{I}}$ for each $A(x) \in q$, and (ii) $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$ for each $r(x,y) \in q$. If such π exists, we write $\mathcal{I} \models q$. We write $\mathcal{K} \models q$ (and say "$\mathcal{K}$ entails q") if there is a match for q in every model of the KB $\mathcal{K}$. This defines the problem that we consider in this paper: given a KB $\mathcal{K}$ and a CQ q, decide whether $\mathcal{K}$ entails q. There is one additional simplifying assumption concerning the structure of the query. As usual and w.l.o.g., we assume throughout the paper that all queries are *connected*, i.e., there is only one connected component in the *query graph* G^q, which is the directed graph with nodes $\mathbf{V}(q)$ that has an edge from x to y iff $r(x,y) \in q$ for some role r. Non-connected queries can be answered by independently answering each connected component.

In most parts of this paper, we will concentrate on knowledge bases that contain only a single concept assertion $C_0(a_0)$ and no role assertions. We call such a KB *simple*. Just like the simplifying assumptions that we make on conjunctive queries, this serves the purpose of an easier exposition. In contrast to those assumptions, however, this case *is* technically simpler than the general case. We will discuss this in more detail in Section 3.4. When deciding query entailment over a simple $\mathcal{ALCHI}$ KB $\mathcal{K}$, it suffices to concentrate on models of $\mathcal{K}$ that are tree shaped. Formally, a model $\mathcal{I}$ of $\mathcal{K}$ is a *tree model* if:

1. $\Delta^{\mathcal{I}}$ is a prefixed-closed subset of $\mathbb{N}^*$ (i.e., of words over the natural numbers); the empty word is denoted by ε and called the *root* of $\Delta^{\mathcal{I}}$,
2. if $(d,e) \in (\exists r.C)^{\mathcal{I}}$ for some role r and some concept C, then, for some $j \in \mathbb{N}$, $(d, d{\cdot}j) \in r^{\mathcal{I}}$ and $d{\cdot}j \in C^{\mathcal{I}}$, and

3. if $(d, e) \in r^{\mathcal{I}}$ for some role name r, then, for some $j \in \mathbb{N}$, either (i) $e = d{\cdot}j$, or (ii) $d = e{\cdot}j$.

If $\mathcal{I}$ is a tree model of $\mathcal{K}$, and for each $(d, e) \in r^{\mathcal{I}}$ we have $e = d{\cdot}j$ for some $j \in \mathbb{N}$ (i.e., if (3ii) never applies), then $\mathcal{I}$ is a *1-way tree model*. In the case of $\mathcal{ALCH}$ KBs, it even suffices to concentrate on 1-way tree models. From now on, the *size* of a knowledge base $\mathcal{K}$ is denoted $|\mathcal{K}|$.

Proposition 1. *Every consistent simple $\mathcal{ALCHI}$ KB $\mathcal{K}$ has a tree model, and for each query q, if $\mathcal{K} \not\models q$, then there exists a tree model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. The model $\mathcal{I}$ is such that $\Delta^{\mathcal{I}} \subseteq \{0, \ldots, |\mathcal{K}| - 1\}^*$ and it is 1-way if $\mathcal{K}$ is an $\mathcal{ALCH}$ KB.*

We note that a similar proposition can be established for non-simple KBs, but then tree models have to be replaced by forest-like ones; see Section 3.4.

3 Query Answering by Knot Elimination

We describe the knot technique, and show how it can be used to decide CQ entailment over simple KBs formulated in $\mathcal{ALCH}$ and $\mathcal{ALCHI}$, yielding tight upper complexity bounds. Then we discuss extensions to general KBs and more expressive DLs.

3.1 Knots

The aim of the *knot technique* is to obtain a finite representation of potentially infinite tree models by decomposing them into a collection of small pieces. Each such piece is described by a *knot*, a schematic labeled tree of depth ≤ 1 with bounded branching. A knot describes a node in the tree model together with all its successors, fixing the concepts that are satisfied at each node and the roles connecting the nodes. By restricting ourselves to only the *relevant* concepts and roles, we achieve that only finitely many distinct knots exist, and thus every tree model can be represented as a finite knot set. Conversely, knots can be viewed as the 'building blocks' for a potential tree model. To ensure that knots can indeed be assembled into such a model, two kinds of conditions are imposed. *Local* conditions apply to individual knots and deal with the internal consistency of the nodes in a knot. *Global* conditions ensure that instances of the knots in a set can be assembled together.

We assume from now on that concepts are in negation normal form (NNF). It is well know that every concept can be converted into an equivalent one in NNF in linear time. We use $\sim C$ to denote the result of converting $\neg C$ into NNF. For the rest of the section, fix a simple $\mathcal{ALCHI}$ knowledge base $\mathcal{K}$ with concept atom $C_0(a_0)$.

Definition 1 (Knot). *Let $\mathsf{cl}(\mathcal{K})$ denote the smallest set of concepts that contains every concept in $\mathcal{K}$ and is closed under subconcepts and NNF-negation "$\sim$". A* concept-type *for $\mathcal{K}$ is a set $\tau \subseteq \mathsf{cl}(\mathcal{K})$ such that for all $C, D \in \mathsf{cl}(\mathcal{K})$,*

1. $C \sqsubseteq D \in \mathcal{K}$ *implies* $\sim C \in \tau$ *or* $D \in \tau$;
2. $C \in \tau$ *implies* $\sim C \notin \tau$,
3. *if* $C \sqcap D \in \tau$, *then* $\{C, D\} \subseteq \tau$, *and*
4. *if* $C \sqcup D \in \tau$, *then* $C \in \tau$ *or* $D \in \tau$.

A role-type (for $\mathcal{K}$) *is a set* $\rho \subseteq \mathbf{R}_\mathcal{K} \cup \{p^- \mid p \in \mathbf{R}_\mathcal{K}\}$. *A* knot for $\mathcal{K}$ *is a pair* $\kappa = (\tau, S)$ *that consists of a concept-type* τ *for* $\mathcal{K}$ *(called* root type*) and a set* S *of pairs* (ρ, τ'), *where* ρ *and* τ' *are a role-type and a concept-type for* $\mathcal{K}$.

A knot $\kappa = (\tau, S)$ can be viewed as a tree of depth ≤ 1, whose nodes are labeled with subsets of $\mathsf{cl}(\mathcal{K})$, and whose edges are labeled with sets of roles. More specifically, τ is the label of the root node and each pair $(\rho, \tau') \in S$ describes a successor with edge label ρ and node label τ'. Next, we define local conditions for knots which ensure that there are no contradictions within the knot.

Definition 2 (Knot consistency). *A knot* $\kappa = (\tau, S)$ *is* $\mathcal{K}$-consistent *if:*

1. *if* $\exists r.C \in \tau$, *then* $r \in \rho$ *and* $C \in \tau'$ *for some* $(\rho, \tau') \in S$;
2. *if* $\forall r.C \in \tau$, *then* $C \in \tau'$ *for all* $(\rho, \tau') \in S$ *with* $R \in \rho$;
3. *if* $\forall r.C \in \tau'$ *for some* $(\rho, \tau') \in S$ *and* $r^- \in \rho$, *then* $C \in \tau$; *and*
4. *if* $r \sqsubseteq s \in \mathcal{K}$ *and* $(\rho, \tau') \in S$, *then* $r \in \rho$ *implies* $s \in \rho$, *and* $r^- \in \rho$ *implies* $s^- \in \rho$.
5. $|S| \leq |\mathsf{cl}(\mathcal{K})|$.

A knot being free of local contradictions does not yet guarantee that it can be part of a tree model as there could be existential restrictions at successor nodes that cannot be expanded into a full model. We therefore also need a global condition which guarantees that such an expansion is always possible.

Definition 3 (Coherency of knot sets). *Given a knot set* $\mathbb{K}$, *a knot* $(\tau, S) \in \mathbb{K}$ *is* good in $\mathbb{K}$, *if for each* $(\rho, \tau') \in S$, *there is a knot* $(\tau_s, S_s) \in \mathbb{K}$ *with* $\tau' = \tau_s$. *Then* $\mathbb{K}$ *is* $\mathcal{K}$-coherent *if (i) each knot* $(\tau, S) \in \mathbb{K}$ *is* $\mathcal{K}$*-consistent and good in* $\mathbb{K}$; *and (ii) there is a knot* $(\tau, S) \in \mathbb{K}$ *with* $C_0 \in \tau$.

A tree-shaped model $\mathcal{I}$ of $\mathcal{K}$ can be decomposed into a $\mathcal{K}$-coherent knot set in a straightforward way. Conversely, if we have a $\mathcal{K}$-coherent set of knots $\mathbb{K}$, we can build a tree model of $\mathcal{K}$: start with the knot (τ, S) with $C_0 \in \tau$ as the 'root knot', then repeatedly append suitable successor knots to the leafs of the tree.

Theorem 1. $\mathcal{K}$ *is consistent iff there exists a* $\mathcal{K}$*-coherent knot set.*

3.2 Non-entailment of a Set of Tree-Shaped Queries

We now present a knot-based approach to decide query entailment in $\mathcal{ALCH}$ and $\mathcal{ALCHI}$, which yields a tight ExpTime upper bound in the $\mathcal{ALCH}$ case, and a tight 2-ExpTime upper bound for $\mathcal{ALCHI}$. We proceed in two steps. The first step is presented in the current section, where we give a knot-based algorithm for query entailment in $\mathcal{ALCHI}$ that presupposes tree-shaped queries and runs in ExpTime. In fact, the algorithm works with sets of queries and decides a special, non-standard version of entailment. The second step is presented in the

subsequent section, where we reduce standard query entailment to the special case treated in the current section.

We say that a query q is *tree-shaped* if the query graph G^q is a tree. We assume that the nodes in tree-shaped queries q have canonical names: the root of G^q is x_ϵ and if x_w is a node in G^q with n children, then these children are $x_{w\cdot 1}, \ldots, x_{w\cdot n}$. For a variable x_w, we denote by $\mathsf{subq}(q, x_w)$ the canonical query obtained by restricting q to the subtree rooted at x_w, renaming the nodes accordingly. We now introduce the special kind of entailment used in this section.

Definition 4 (Directed Entailment). *Let $\mathcal{K}$ be an $\mathcal{ALCHI}$ KB, $\mathcal{I}$ be a tree model of $\mathcal{K}$, Q a set of tree-shaped queries, and $q \in Q$. For $d \in \Delta^{\mathcal{I}}$, we write $\mathcal{I} \models^* q[d]$ if there exists a match π for q in $\mathcal{I}$ such that $\pi(x_\epsilon) = d$ and for every $r(x, y) \in q$, we have $\pi(y) = \pi(x)\cdot i$ for some $i \geq 0$. We write $\mathcal{I} \models^* q$ if $\mathcal{I} \models^* q[d]$ for some $d \in \Delta^{\mathcal{I}}$, $\mathcal{I} \models^* Q$ if $\mathcal{I} \models^* q$ for some $q \in Q$, and $\mathcal{K} \models^* Q$ if $\mathcal{I} \models^* Q$ for every tree model $\mathcal{I}$ of the knowledge base $\mathcal{K}$.*

Observe that the matches used in Definition 4 are directed in the sense that, even if the tree model is not 1-way, we map every atom $r(x, y) \in q$ only "downwards". If the tree model is 1-way *and* the query does not involve inverse roles as in the case of $\mathcal{ALCH}$, then these matches coincide with standard ones, i.e., we have $\mathcal{K} \models q$ iff $\mathcal{K} \models^* \{q\}$.

The algorithm devised in this section decides, given an $\mathcal{ALCH}$ or $\mathcal{ALCHI}$ KB $\mathcal{K}$ and a set of tree-shaped queries Q, whether $\mathcal{K} \models^* Q$. The following characterization of directed entailment is easy to establish.

Proposition 2. *Let $\mathcal{I}$ be a tree model of an $\mathcal{ALCHI}$ KB $\mathcal{K}$, $d \in \Delta^{\mathcal{I}}$, and q a tree-shaped query. Then $\mathcal{I} \not\models^* q[d]$ iff one of the following holds:*

(i) $\{A \mid A(x_\epsilon) \in q\} \not\subseteq \{A \mid d \in A^{\mathcal{I}}\}$ *or*
(ii) there exists a variable $x_{\epsilon\cdot i}$ of q such that for each child $d\cdot j \in \Delta^{\mathcal{I}}$ of d we have:
- $\{r \mid r(x_\epsilon, x_{\epsilon\cdot i}) \in q\} \not\subseteq \{r \mid (d, d\cdot j) \in r^{\mathcal{I}}\}$, *or*
- $\mathcal{I} \not\models^* \mathsf{subq}(q, x_{\epsilon\cdot i})[d\cdot j]$.

For the rest of the section, fix a simple $\mathcal{ALCH}$ or $\mathcal{ALCHI}$ KB $\mathcal{K}$ and a set of tree-shaped queries Q. The aim of our algorithm is to decide whether $\mathcal{K} \models^* Q$ by using knots to verify the existence of a tree model $\mathcal{I}$ of $\mathcal{K}$ with $\mathcal{I} \not\models^* Q$ (a *countermodel*). Proposition 2 suggests that we can do this by extending knots with auxiliary information that enables us to track the satisfaction of conditions (i) and (ii) for each query in Q at each node of the tree.

Definition 5 (Marked knots). *Let Q^* denote the smallest set such that $Q \subseteq Q^*$, and if $q \in Q^*$ and $x_{\epsilon\cdot i}$ is a variable of q, then $\mathsf{subq}(q, x_{\epsilon\cdot i}) \in Q^*$. A Q-*marked *knot is a tuple (τ, S, ν) where (τ, S) is a knot and $\nu : \{\epsilon\} \cup S \to 2^{Q^*}$.*

Intuitively, every node in a Q-marked knot is labeled with the set of those subqueries of queries in Q for which a (directed) match of the root should be *avoided* at that node. To capture this formally, we define additional local conditions.

algorithm CounterModel($\mathcal{K}, Q$)
Compute the set $\mathbb{K}_0$ of all Q-avoiding knots for $\mathcal{K}$
$i := 0$
repeat
  $i := i + 1$
  $\mathbb{K}_i := \mathbb{K}_{i-1} \setminus \{(\tau, S, \nu) \in \mathbb{K}_{i-1} \mid (\tau, S, \nu)$ is not good in $\mathbb{K}_{i-1}\}$
until $\mathbb{K}_i \neq \mathbb{K}_{i-1}$;
if there is $(\tau, S, \nu) \in \mathbb{K}_i$ with $C_0 \in \tau$ **then return** " a counter model exists"
  else return "a counter model does not exist"

Fig. 1. The knot elimination algorithm

Definition 6 (Query avoiding knots). *A Q-marked knot (τ, S, ν) is* Q-avoiding*, if for each $q \in Q$, we have $q \in \nu(\epsilon)$ and that one of the following holds:*

(a) $\{A \mid A(x_\epsilon) \in q\} \not\subseteq \tau$*, or*
(b) there exists some variable $x_{\epsilon \cdot i}$ such that for every $(r, \tau') \in S$, it holds that $\{r \mid r(x_\epsilon, x_{\epsilon \cdot i}) \in q\} \not\subseteq \rho$ *or* $\mathsf{subq}(q, x_{\epsilon \cdot i}) \in \nu((\rho, \tau))$.

The above just mimics the conditions in Proposition 2. We now define global conditions to ensure that the marking is consistent between knots.

Definition 7 (Coherency of marked knot sets). *For a set $\mathbb{K}$ of Q-marked knots, we call $(\tau, S, \nu) \in \mathbb{K}$* good in $\mathbb{K}$ *if for each $(\rho, \tau') \in S$, there is some $(\tau_s, S_s, \nu_s) \in \mathbb{K}$ such that $\tau' = \tau_s$ and $\nu((\rho, \tau')) = \nu_s(\epsilon)$. Then $\mathbb{K}$ is* $\mathcal{K}$-coherent *if for each $(\tau, S, \nu) \in \mathbb{K}$, (τ, S) is $\mathcal{K}$-consistent and (τ, S, ν) is Q-avoiding and good in $\mathbb{K}$.*

To show the following, we can now argue as for Theorem 1, additionally using Proposition 2.

Proposition 3. *$\mathcal{K} \not\models^* Q$ iff there exists a $\mathcal{K}$-coherent set of Q-marked knots.*

To decide whether $\mathcal{K} \not\models^* Q$, it thus suffices to decide the existence of a knot set as in Proposition 3. This is done by knot elimination, inspired by the so-called type elimination technique due to Pratt; see Section 4. The details of the algorithm are presented in Figure 1, where we assume that $C_0(a_0)$ is the concept assertion in $\mathcal{K}$. The algorithm runs in exponential time since there are only exponentially many Q-marked knots for $\mathcal{K}$ and it can be checked in polynomial time whether a knot is Q-avoiding and good in a knot set.

Theorem 2. *Given $\mathcal{K}$ and Q, it can be decided in time exponential in the size of $\mathcal{K}$ and Q whether $\mathcal{K} \models^* Q$.*

3.3 From Unrestricted Queries to Tree-Shaped Ones

We now show how Theorem 2 can be used to derive tight complexity bounds for standard entailment of a conjunctive query q that is not necessarily tree-shaped by a simple $\mathcal{ALCHI}$ KB $\mathcal{K}$. By Proposition 1, q not being entailed by $\mathcal{K}$ implies that there is a tree model $\mathcal{I}$ of $\mathcal{K}$ that witnesses non-entailment. Further, any match π for q in model $\mathcal{I}$ gives rise to a rewriting q_π of q as follows:

- the variables of q_π are $\{x_d \mid \exists x \in \mathbf{V}(q) : \pi(x) = d\}$;
- the concept atoms of q_π are $\{A(x_d) \mid \exists\ A(x) \in q : \pi(x) = d\}$;
- the role atoms are $\{r(x_d, x_{d\cdot i}) \mid \exists\ r(x, y) \in q : \pi(x) = d \text{ and } \pi(y) = d \cdot i\} \cup \{r(x_d, x_{d\cdot i}) \mid \exists\ r^-(y, x) \in q : \pi(x) = d \text{ and } \pi(y) = d \cdot i\}$

Since q is connected and $\mathcal{I}$ is a tree model, q_π is obviously tree-shaped (and it is straightforward to assign canonical names to the variables). Moreover, the construction of q_π ensures that $\mathcal{I} \models^* q_\pi$, i.e., there is a *directed* match for q_π in $\mathcal{I}$. This observation suggests that entailment of q can be verified by replacing q with a set of tree-shaped rewritings and checking directed entailment.

Definition 8. *A tree-shaped query q' is a* tree rewriting *of q if there is a surjective map $\nu : \mathbf{V}(q) \to \mathbf{V}(q')$ such that*

(i) $A(x) \in q$ *iff* $A(\nu(x)) \in q'$, *and*
(ii) $r(x, y) \in q$ *iff* $r(\nu(x), \nu(y)) \in q'$ *or* $r^-(\nu(y), \nu(x)) \in q'$.

Let $\mathsf{TRew}(q)$ *denote all tree rewritings of q.*

Based on the observation above, the following is easy to prove.

Lemma 1. *For each simple $\mathcal{ALCHI}$ KB $\mathcal{K}$ and CQ q, we have $\mathcal{K} \not\models q$ iff $\mathcal{K} \not\models^* \mathsf{TRew}(q)$.*

For an $\mathcal{ALCHI}$ KB $\mathcal{K}$, we can thus decide whether $\mathcal{K} \models q$ by using the algorithm from the previous section with $Q = \mathsf{TRew}(q)$. Since the cardinality of $\mathsf{TRew}(q)$ is exponential in the size of q, we obtain a 2-ExpTime upper bound. This bound is tight: in [12], it was shown that CQ entailment over simple $\mathcal{ALCI}$ KBs (i.e., $\mathcal{ALCHI}$ KBs without role inclusions) is 2-ExpTime-hard.

In the case of $\mathcal{ALCH}$, we can replace $\mathsf{TRew}(q)$ with a single query! Recall that $\mathcal{ALCH}$ terminologies enjoy 1-way tree models and that we disallow inverse roles in the query when working with $\mathcal{ALCH}$. Together, this means that if we have $\mathcal{I} \models q$ with $\mathcal{I}$ a 1-way tree-model of the (simple) input KB $\mathcal{K}$, we can obtain a tree-shaped rewriting q' of q with $\mathcal{I} \models q'$ in a very easy way: simply eliminate all *forks* $r(x, y), r(x', y)$ in q by identifying the variables x and x'. Observe that, in contrast to the case of $\mathcal{ALCHI}$, this rewriting is independent of the concrete match π of q in $\mathcal{I}$. Thus, we obtain only a single rewriting q' (which can be obtained in polynomial time). As noted already in Section 3.2, we then have $\mathcal{I} \models q'$ iff $\mathcal{I} \models^* q'$ which enables the use of the algorithm in the previous section.

Definition 9 (Query rewriting). *[12] Let $\mathcal{K}$ be a simple $\mathcal{ALCH}$ KB, q a CQ without inverse roles, and let* $\mathsf{FE}(q)$ *denote the result of eliminating all forks in q. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models^* \{\mathsf{FE}(q)\}$.*

We thus obtain an ExpTime upper bound by Theorem 2. A lower bound is easily obtained by a trivial reduction of satisfiability in $\mathcal{ALCH}$.

3.4 Extensions

The knot-based approach to query answering can be extended to the case of non-simple KBs and to more expressive DLs than $\mathcal{ALCH}$ and $\mathcal{ALCHI}$. We start with the former. As noted in Section 2, Proposition 1 can be adapted to the case of non-simple KBs by replacing tree models with forest-shaped ones. More precisely, such models consist of a core whose relational structure is unrestricted and a collection of possibly infinite trees whose roots are from the core. The core contains precisely those elements that are identified by some individual name in the knowledge base $\mathcal{K}$, and thus its size is bounded by that of $\mathcal{K}$. This also explains why the relational structure of the core cannot be restricted: it needs to mirror the role assertions $r(a, b)$ in $\mathcal{K}$.

In the knot approach, the whole core is represented by a single, large knot, cf. the *min-graphs* of [20]. To avoid matches when constructing a countermodel, we now have to deal with three types of matches: (i) matches located purely inside the core; (ii) matches located partially in the core and partially in one or more of the trees; and (iii) matches located purely inside a tree. This can be done by a careful extension of the local and global conditions for marked knots and is most subtle in the case of $\mathcal{ALCH}$. There, it is crucial to show that, although in matches of type (ii) there are exponentially many ways to split the query between the core and the trees, only polynomially many queries need to be taken into account when avoiding matches in the tree parts [20,13]. The assumption made in this paper that knowledge bases are simple allows us to concentrate on matches of type (iii). Though this may appear to be the easiest of the three cases, in $\mathcal{ALCH}$ and $\mathcal{ALCHI}$ it is actually the source of complexity: the 2-ExpTime lower bound for $\mathcal{ALCHI}$ in [12] applies to simple KBs, and the complexity does not increase for non-simple ones [9].

We now come to more expressive DLs than $\mathcal{ALCH}$ and $\mathcal{ALCHI}$. It is not hard to extend our approach to *number restrictions* by imposing counting constraints on the successors in knots, thus capturing the DLs $\mathcal{ALCHQ}$ and $\mathcal{ALCHIQ}$. The extended algorithm still yields ExpTime and 2-ExpTime upper bounds, respectively, though some care has to be taken when combining $\mathcal{Q}$ and $\mathcal{I}$. A particularly interesting extension is provided by *transitive roles*. In their presence, it is not possible to generate a set of tree-shaped queries of polynomial size even when we are working with 1-way models and simple KBs. The knot-based approach thus yields a 2-ExpTime upper bound [19], which is optimal if the considered DL also has role inclusions (i.e., if it contains the DL $\mathcal{SH}$) [5]. In [19], a class of queries is identified for which the complexity of CQ entailment in $\mathcal{SH}$ drops to ExpTime. When we disallow role inclusions, the 2-ExpTime upper bound is no longer tight. In fact, a somewhat intricate refinement of the knot approach can be used to show that, in $\mathcal{ALC}$ with transitive roles (also known as $\mathcal{S}$), CQ entailment w.r.t. simple KBs is still in ExpTime [5]. In the same paper, we show that for non-simple KBs, the complexity raises to co-NExpTime-hardness. In a nutshell, this is due to matches of type (ii) being more complicated than without transitive roles and giving rise to an exponential number of queries to be avoided in the tree parts of models. A tight complexity bound is still missing.

4 Related Work and Conclusion

There is a large number of other approaches to conjunctive query entailment in DLs, including reductions to satisfiability [9,8,12], automata and tableaux methods [3,18], and resolution [11]. We omit a detailed discussion due to lack of space and instead discuss techniques that are similar in spirit to knots. As already mentioned, knots are a special instance of the *mosaic* technique [17] that has been used to obtain decidability and complexity results in modal and description logic. The reader may refer to [16,2] and consult e.g. [14,24] as examples from the DL literature. With the exception of [4,23], we are not aware of papers in which other variations of the mosaic technique have been used for query answering. Both knots and mosaics are closely related to *type elimination*, which has been used extensively in description and modal logic, see e.g. [22,21,10,15]. Roughly, a type is a small mosaic with only one element and type elimination is the analogue of Figure 1 with knots replaced by types. We remark that the algorithm in Section 3.2 can also be formulated using annotated types instead of annotated knots. However, using knots allows for simpler local and global conditions, especially when extending the approach to more expressive DLs such as those involving transitive roles.

Summing up, in this paper we have illustrated how the knot technique can be applied for answering conjunctive queries in DLs. The method is conceptually simple yet powerful enough to handle different DLs with considerably different computational properties. To wit, we presented a worst-case optimal algorithm that directly scales from the DL $\mathcal{ALCH}$ to the exponentially harder DL $\mathcal{ALCHI}$. Given that knots are special mosaics tailored for DLs with tree-shaped models, investigating the mosaic technique for query answering in more expressive DLs which lack this property, like (fragments of) $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, is an interesting topic for future research.

References

1. Baader, F., Lutz, C., Motik, B. (eds.): Proceedings of the 21st International Workshop on Description Logics (DL 2008), Dresden, Germany, May 13-16, 2008. CEUR Workshop Proceedings, vol. 353. CEUR-WS.org (2008)
2. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Sc., vol. 53. Cambridge University Press, Cambridge (2001)
3. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI 2007), pp. 391–396 (2007)
4. Eiter, T., Gottlob, G., Ortiz, M., Šimkus, M.: Query answering in the description logic Horn-SHIQ. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS, vol. 5293, pp. 166–179. Springer, Heidelberg (2008)
5. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in description logics with transitive roles. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). AAAI Press/IJCAI (2009)
6. Eiter, T., Ortiz, M., Simkus, M.: Reasoning using knots. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS, vol. 5330, pp. 377–390. Springer, Heidelberg (2008)

7. Fox, D., Gomes, C.P. (eds.): Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008. AAAI Press, Menlo Park (2008)
8. Glimm, B., Horrocks, I., Sattler, U.: Conjunctive query entailment for $\mathcal{SHOQ}$. In: Proc. of the 2007 Description Logic Workshop (DL 2007). CEUR Electronic Workshop Proceedings, vol. 250, pp. 65–75 (2007), http://ceur-ws.org/Vol-250/
9. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. Journal of Artificial Intelligence Research 31, 150–197 (2008)
10. Halpern, J.Y., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artif. Intell. 54(3), 319–379 (1992)
11. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS, vol. 3452, pp. 21–35. Springer, Heidelberg (2005)
12. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
13. Lutz, C.: Two upper bounds for conjunctive query answering in SHIQ. In: Baader, et al [1]
14. Lutz, C., Sattler, U., Tendera, L.: The complexity of finite model reasoning in description logics. Inf. Comput. 199(1-2), 132–171 (2005)
15. Lutz, C., Wolter, F., Zakharyaschev, M.: Temporal description logics: A survey. In: Demri, S., Jensen, C.S. (eds.) Proc. 15th International Symposium on Temporal Representation and Reasoning (TIME 2008), pp. 3–14. IEEE Computer Society, Los Alamitos (2008)
16. Marx, M., Venema, Y.: Local variations on a loose theme: Modal logic and decidability. In: Finite Model Theory and Its Applications, vol. 7, pp. 371–429. Springer, Heidelberg (2007)
17. Németi, I.: Free algebras and decidability in algebraic logic. DSc. thesis, Mathematical Institute of The Hungarian Academy of Sciences, Budapest (1986)
18. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. J. of Automated Reasoning 41(1), 61–98 (2008)
19. Ortiz, M., Simkus, M., Eiter, T.: Conjunctive query answering in SH using knots. In: Baader, et al [1]
20. Ortiz, M., Simkus, M., Eiter, T.: Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In: Fox and Gomes [7], pp. 504–510
21. 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)
22. Pratt, V.R.: Models of program logics. In: FOCS, pp. 115–122. IEEE, Los Alamitos (1979)
23. Pratt-Hartmann, I.: Data-complexity of the two-variable fragment with counting quantifiers. Information and Computation (2008) (Forthcoming)CoRR, http://arxiv.org/abs/0806.1636
24. Rudolph, S., Krötzsch, M., Hitzler, P.: Terminological reasoning in $\mathcal{SHIQ}$ with ordered binary decision diagrams. In: Fox and Gomes [7], pp. 529–534

Mathematical Logic for Life Science Ontologies

Carsten Lutz[1] and Frank Wolter[2]

[1] Department of Computer Science, University of Bremen, Germany
clu@informatik.uni-bremen.de
[2] Department of Computer Science, University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract. We discuss how concepts and methods introduced in mathematical logic can be used to support the engineering and deployment of life science ontologies. The required applications of mathematical logic are not straighforward and we argue that such ontologies provide a new and rich family of logical theories that wait to be explored by logicians.

1 Introduction

In recent years, life sciences such as medicine and biology have experienced an abundant growth of terminology. This is mainly due to increasing interdisciplinarity and significant scientific advances such as the decoding of the human genome, which have established novel research areas such as pharmacogenomics (study of the effect of genes on drug response). Indeed, life science terminology has grown to a point where it information processing and exchange is seriously hampered as it can no longer be guaranteed that multiple parties interpret the data in the same way and using the same terminology. To address this problem, there is a strong trend to construct reference terminologies, which list a standard vocabulary to be used in information processing and data exchange, and which also fixes the meaning of each vocabulary item.

Reference terminologies, which are nowadays usually called *ontologies*, are often given in a logical language to obtain a formal semantics and make use of automated reasoning technology. More specifically, a logical signature is used to define the vocabulary and a (finitely axiomatized) logical theory defines the meaning of the vocabulary. Such ontologies are typically formulated in fragments of first-order logic and are thus amenable to the concepts and tools developed in mathematical logic. The aim of this paper is to provide a glimpse into the field of life science ontologies for readers with a mathematical logic background, to demonstrate that concepts and tools from mathematical logic can be fruitfully applied to engineering problems for ontologies, and to argue that, conversely, life science ontologies provide a rich landscape of logical theories that present interesting and novel challenges for logicians.

The paper is divided into two parts. In the first part, we introduce a typical and successful example of a medical ontology: SNOMED CT, the Systematized Nomenclature of Medicine, Clinical Terms. We give an idea of what SNOMED CT is, how it is developed and kept up to date, what its main applications are, and

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 37–47, 2009.

what kind of logical axioms are used to define the logical theory unterlying it. In the second part of this paper, we show how concepts from mathematical logic can be applied to ontologies in general, and to SNOMED CT in particular. As concrete examples for successful such applications, we consider conservative extensions as a tool for achieving modularity of ontologies and uniform interpolants as a tool for obtaining an axiomation of a relevant fragment of an ontology. We close with some hints to other techiques from mathematical logic that potentially have interesting applications in the area of ontologies.

2 SNOMED CT

A large number of medical ontologies have emerged in recent years, including the National Cancer Institute (NCI) thesaurus [11], the Foundational Model of Anatomy (FMA) [1], Galen [26], and SNOMED CT [22,28]. Among these ontologies, SNOMED CT plays a particularly prominent role as it is used to produce the standard healthcare terminology for a number of countries such as the US, Canada, the UK, and Sweden.

2.1 Overview

SNOMED CT is a comprehensive clinical healthcare terminology that comprises more than 400.000 vocabulary items and almost the same number of logical axioms. Since 2007, the intellectual property rights of SNOMED CT are held by a not-for-profit association called International Heath Terminology Standards Development Organisation (IHSTDO). Currently, IHSTDO is made of nine nations, including the ones listed above. The general goal is to develop SNOMED CT into *the* global clinical terminology, thus "enabling clinicians, researchers and patients to share and exchange healthcare and clinical data worldwide" [2].

Technically, SNOMED CT is a finite set of first-order predicate logic sentences and uses standard Tarski semantics. Medical terms are regarded as unary predicates such as

$$\mathsf{Disease}, \quad \mathsf{Appendicitis}, \quad \mathsf{Inflammation}, \quad \mathsf{Arthritis},$$

and binary predicates such as

$$\mathsf{Associated\_Morphology}, \quad \mathsf{Finding\_Site}, \quad \mathsf{Procedure\_Site}.$$

An example of a typical SNOMED CT axiom is as follows:

$$\forall x\, (\mathsf{Appendicitis}(x) \rightarrow \mathsf{Disease}(x) \wedge \exists y(\mathsf{Associated\_Morphology}(x,y) \wedge \mathsf{Inflammation}(y))).$$

Though easily translatable, the official syntax of SNOMED CT is not classical first-order syntax and does not contain variables and first-order quantifiers. Instead, the syntax is based on so-called description logics (DLs), a family of

knowledge representation formalisms often used as ontology languages [6]. In description logic, a theory generally consists of axioms of the form

$$\forall x\ (\varphi(x) \rightarrow \psi(x)) \quad \text{and} \quad \forall x\ (\varphi(x) \leftrightarrow \psi(x)),$$

where $\varphi(x)$ and $\psi(x)$ are first-order formulas with one free variable using unary and binary predicates (formulated in DL syntax).

The admissible form of $\varphi(x)$ and $\psi(x)$ depends on the description logic used. Without going into any details, we briefly give three examples of descriptions logics in increasing order of expressivity. The most inexpressive one is $\mathcal{EL}$ [5], where $\varphi(x)$ and $\psi(x)$ are composed using conjunction and guarded existential quantification. Although inexpressive, $\mathcal{EL}$ is a popular ontology language and is used, for example, for SNOMED CT. When $\mathcal{EL}$ is extended with disjunction, negation, and guarded universal quantification, we obtain the description logic $\mathcal{ALC}$. For example, the NCI thesaurus is formulated in this language. By further admitting guarded counting quantifiers and a number of other constructors, one obtains the description logic OWL DL (the Web Ontology Language) that, for example, is used for the medical ontology GALEN.

In many ontologies, the structure of axioms is restricted further by imposing *unfoldability*. In this case, $\varphi(x)$ has to be an atom $P(x)$, and thus axioms $\forall x\ (P(x) \rightarrow \psi(x))$ give necessary conditions for being a P and axioms $\forall x\ (P(x) \leftrightarrow \psi(x))$ give both necessary and sufficient conditions. In addition, no predicate name may be used on the left-hand side of more than one axiom and an acyclicity condition is enforced which basically says that ψ may not refer to P, neither directly nor indirectly. Both SNOMED CT and NCI are unfoldable terminologies, whereas GALEN is not.

2.2 Applications and Engineering

The main purpose of SNOMED CT is to produce a hierarchically organized catalogue of medical terms, with more general terms higher up in the hierarchy and more specialized ones further down. Each term is annotated with a natural language description of its meaning, a numerical code that uniquely identifies the term, and a logical axiom that defines the term's meaning (on a high level of abstraction).

The classical application of SNOMED CT is to simply use the catalogue and numerical codes in medical IT applications, without exploiting or having access to the logical axioms [3] (but see below). As an example for this use of SNOMED CT, consider the generation, processing, and storage of medical records. Whether working in a hospital or independently, physicians have to generate a detailed record of every patient visit, including details on findings, diagnosis, treatment, and medication. The purpose of the resulting medical records is manifold and ranges from archiving via accounting and billing to communication with external labs and hospitals. When a physician enters a medical record at his PC, he can browse medical terms by navigating along the SNOMED CT hierarchy and view the natural language descriptions when necessary. After selecting a relevant term,

the SNOMED CT numerical code can be automatically inserted into the record. In fact, there is a strong trend towards *electronic* medical records, and standard data formats such as the Health Level 7 Clinical Document Architecture (CDA) are already in wide-spread use. To make sure that medical records represented in such standard formats are not misinterpreted, it is important to use standardized medical terminology. To this effect, CDA is based on SNOMED CT numerical codes (among others). The general idea of SNOMED CT is that, if this standard is adopted by the healthcare system of a nation, then *all* medical data storage and exchange is in terms of SNOMED CT codes.

Why, then, is SNOMED CT a logical theory? We first note that other standardized medical terminologies such as ICD-10 are *not* based on logic. However, the designers of SNOMED CT find it useful to employ logic during the *design and maintenance* of their ontology. In fact, engineering a large terminology is far from trivial. To generate an ontology of the size of SNOMED CT that covers a range of specialized areas, many medical experts and IT experts have to work together and follow agreed-upon design patterns that guide design decisions. But even with a good design methodology, ensuring that the ontology is of consistent quality and free of mistakes is very difficult. It is here that logic comes into play: the designers of SNOMED CT use automated reasoning to verify their modelling and derive consequences that are only implicit in it. Traditionally, they concentrate on deciding implication in the description logic $\mathcal{EL}$; for example, a recently discovered mistake was the following entailed implication:

$$\text{SNOMED CT} \models \forall x\, (\mathsf{Amputation\_of\_arm}(x) \rightarrow \mathsf{Amputation\_of\_hand}(x)).$$

As we will discuss later, deciding entailment of implications is only the tip of the iceberg and there are many more opportunities for automated reasoning and logic techniques to support ontology design. Additional need for automated reasoning support is due to the fact that medical knowledge, national legislation, etc., are constantly changing, which requires frequent changes to the ontology. Finally, SNOMED CT is customized to national needs and translated into various (human) languages, which results in a large number of different, but closely related versions of the same ontology that need to be consistent with one another. From the perspective of mathematical logic, we thus have a huge and continuously developing first-order theory and a large number of slightly modified variants of this theory that are closely interrelated.

The design of large ontologies is not only difficult, but also time-consuming and expensive. For this reason and since IHSTDO is pleasantly liberal in giving out SNOMED CT for research purposes, SNOMED CT is increasingly being viewed as a valuable general-purpose tool and many novel applications are being proposed. Often, these novel applications involve logical reasoning and provide additional opportunities to apply techniques from logic. We mention only one such application as an example. Given that electronic medical records are gaining rapid popularity and that they use SNOMED CT codes for representing medical data, it is an intriguing idea to exploit not only the codes, but also the *logical definitions* when querying medical data, thus enabling more complete answers. For example, when

answering a query asking for patients with a liver disease, we may use SNOMED CT to deduce that hepatitis is a liver disease and thus include all patients suffering from hepatitis in the answer. In this way, query answering turns into logical deduction. From a logic perspective, an interesting new flavour of this application is scale: in addition to the already large SNOMED CT, the logical theory now also comprises a potentially huge amount of medical data (as ground facts), and new questions arise due to the special use of deduction in this application [23,17].

3 Mathematical Logic

Today, it has become standard to apply automated reasoning and knowledge representation techniques during the design and deployment of ontologies. In particular, reasoning in languages such as $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL has been investigated in depth and found to be decidable and PTIME-, EXPTIME-, and coNEXPTIME-complete, respectively [5,6]. A large variety of automated reasoning systems are available and fruitfully employed by ontology engineers and users. In this section, we consider the potential role of mathematical logic, understood as the study of properties of logical theories and their interrelation. Specifically, we discuss how the notions of conservative extension and uniform interpolation can be employed for ontology engineering tasks. It is worth, though, to mention that these notions have previously been used in other areas of computer science. For example, uniform interpolation has been studied in AI under the name *forgetting* for more than a decade [27,8] and modular program specification is another area of computer science where notions of conservativity and interpolation are of great interest [25,21].

3.1 Conservative Extensions

As mentioned above, ontologies are not static objects, but are frequently corrected, customized, extended, and even merged with other ontologies. To understand and control the relationship between the resulting distinct versions of an ontology, one can directly employ notions from mathematical logic. The simplest such notion is probably that of a conservative extension. In mathematical logic, conservative extensions are used for relative consistency proofs and to decompose a theory into simpler subtheories. In ontology engineering, they can be used to give a rigorous definition of when the extension of an ontology by additional axioms interferes (in some possibly unintended way) with the original ontology, and to decompose an ontology into subtheories and modules [29]. To describe this application in more detail, recall that the notion of a conservative extension comes in two flavours. Let T and T' be two sets of axioms and $\mathcal{L}$ a logical language.

- *Model-theoretic*: $T \cup T'$ is a model-conservative extension of T if every model of T can be expanded to a model of T';
- *Language-dependent*: $T \cup T'$ is an $\mathcal{L}$-conservative extension of T if $T \cup T' \models \varphi$ implies $T \models \varphi$, for all $\mathcal{L}$-sentences φ with $\mathsf{sig}(\varphi) \subseteq \mathsf{sig}(T)$.

In mathematical logic, $\mathcal{L}$ is typically first-order logic (FO) or some language that extends FO. Every model-theoretic conservative extension is an FO-conservative extension, but the converse is well-known to fail. Both notions can, in principle, be used without modification to analyse the effect of adding axioms T' to a given ontology T. It tuns out, however, that the corresponding decision problem "decide whether $T \cup T'$ is a conservative extension of T" is undecidable for theories T and T' formulated in $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL, both for model-conservative extensions and FO-conservative extensions [19,18]. In the case of unfoldable theories, the situation changes: for $\mathcal{ALC}$ and OWL-DL, model and FO-conservativity are still undecidable whereas for $\mathcal{EL}$, model-conservativity is decidable in polynomial time [14].

The tractability result for unfoldable $\mathcal{EL}$-theories has been used to extract modules from SNOMED CT. The need for module extraction is due to the huge size of SNOMED CT and the fact that in many applications, only a small subset Σ of the 400.000 terms of SNOMED CT are required. In this case, it is useful to extract a minimal subset M of the set of SNOMED CT axioms such that Σ is included in the signature $\mathsf{sig}(M)$ of M and SNOMED CT is a conservative extension of M. The user can then work with M instead of SNOMED CT. By extending the algorithm that decides model-conservativity, one can extract such modules in polynomial time. A very encouraging experimental evaluation of this approach has been given in [14], see also the discussion of Figure 3.2 below. Interestingly, the modules extracted using such a logic-based approach are significantly smaller than modules extracted using heuristic approaches.

Under some natural conditions, model-conservativity becomes decidable even for non-unfoldable theories. For example, if T' and T share unary predicates only, then model-conservativity is coNEXPTIME$^{\text{NP}}$-complete for $\mathcal{ALC}$ [14]. Taken together, the stated negative and positive results show that a naive application of mathematical logic concepts to ontology engineering can fail because the associated algorithmic problems become unfeasible even for very weak fragments of FO, and thus the true challenge lies in adapting these notions to the needs of ontology engineers and users. Instead of resorting to unfoldable theories or unary predicates, there is another interesting direction to explore: the reason for undecidability of FO-conservativity in the case of non-unfoldable theories is due to the fact that we have considered *FO*-consequences, rather than consequences formulated at the same level of abstractness as ontologies. Indeed, an ontology engineer or user is typically not interested in arbitrary FO-consequences of an ontology, but in consequences relevant to her application. We now consider two notions of conservative extensions tailored towards ontology engineering.

Implication Conservativity. As illustrated by the "amputation of arm" example above, ontology engineers typically concentrate on implications formulated in the language $\mathcal{L}$ of the ontology. Therefore, it is natural to consider $\mathcal{L}$-conservativity with $\mathcal{L}$ the set of implications $\forall x(\varphi(x) \rightarrow \psi(x))$ formulated in $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL, respectively. For simplicity, we simply speak of $\mathcal{EL}$-conservativity, $\mathcal{ALC}$-conservativity, and OWL-conservativity. To analyze these syntactically defined notions of conservative extension, it is useful to first establish a

model-theoretic characterization. Using techniques from modal logic, one can show that an $\mathcal{ALC}$-theory $T \cup T'$ is an $\mathcal{ALC}$-conservative extension of T if and only if for every model M of T, there exists a $\mathsf{sig}(T)$-bisimilar model M' of $T \cup T'$. Using this characterization, one can show that deciding $\mathcal{ALC}$-conservativity is decidable and 2ExpTime-complete [9]. A similar characterization, using simulations instead of bisimulations, can be used to characterize $\mathcal{EL}$-conservativity. In this case, the corresponding decision problem is ExpTime-complete [19]. Unfortunately, OWL-conservativity is undecidable [12]. We note, though, that there are extensions of $\mathcal{ALC}$ with guarded counting quantifiers for which conservativity is still decidable and 2ExpTime-complete [18].

Because of their high computational complexity, it remains to be seen in how far the decision problems associated with these notions of conservativity are useful in practice. Recall, however, that Snomed CT is unfoldable. To analyze $\mathcal{EL}$-conservativity for unfoldable $\mathcal{EL}$-theories, it is preferable to follow a proof-theoretic approach. Using the sequent calculus of [13], one can give a polynomial time algorithm that decides $\mathcal{EL}$-conservativity. The resulting algorithm has fruitfully been employed to ontology versioning [15].

Query Conservativity. At the end of Section 2, we have briefly discussed how Snomed CT can be used to query electronic medical records. In logic terms, one poses a query to a theory K that consists of an ontology T and a set A of ground facts that represent the data (called an *ABox* in description logic). Query languages of interest are *instance queries* of the form "output all constant symbols a such that $T \cup A \models P(a)$" and *conjunctive queries* in which P is a first-order formula built from conjunction and existential quantification. To compare theories K_1 and K_2, it is in principle possible to again apply standard notions of conservativity. However, ground facts change frequently and are typically unknown at the design time of the ontology. Thus, it is more useful to regard ground facts not as a part of the theory, but as an unknown "black-box". We say that $T \cup T'$ is a *query-conservative extension* of T if, and only if, for all sets A of ground facts and all queries q using symbols from T only, $T \cup T' \cup A \models q[a]$ iff $T \cup A \models q[a]$. The resulting notions of conservativity depend on the query language used and typically lie between implication conservativity and FO-conservativity. Again, model-theoretic and proof-theoretic methods can be applied to analyze it. For example, in [20] it is shown using model-theoretic methods that for $\mathcal{EL}$, query-conservativity (w.r.t. conjunctive queries) can be reduced to implication conservativity for an extension of $\mathcal{EL}$ with non-guarded existential quantification, and that the corresponding decision problem is decidable and ExpTime-complete.

3.2 Uniform Interpolation

As mentioned above, many applications of ontologies require only a rather small part of a large ontology, identified by a subvocabulary. Therefore, ontology users are interested in generating small ontologies that "say the same" about the subvocabulary of interest as the original ontology. As discussed above, one possibility to obtain such an ontology is to extract a module, i.e., a subset of the original

ontology of which the latter is a conservative extension. Another interesting way of generating such a small ontology is to compute a uniform interpolant. Let Σ be a signature, T a logical theory, and $\mathcal{L}$ a logical language. A finite set of $\mathcal{L}$-sentences T_Σ is called a *uniform interpolant* for T w.r.t. Σ and $\mathcal{L}$ if

- the signature $\mathsf{sig}(T_\Sigma)$ of T_Σ is contained in Σ;
- $T \models T_\Sigma$;
- for all $\varphi \in \mathcal{L}$: if $T \models \varphi$ and $\mathsf{sig}(\varphi) \cap T \subseteq \Sigma$, then $T_\Sigma \models \varphi$.

In other words, T_Σ provides an axiomatization of what T "says" about Σ in $\mathcal{L}$ without using symbols not in Σ. Apart from the motivation discussed above, there are various applications of uniform interpolants in ontology engineering. An example is ontology exploration: to avoid mistakes, an ontology engineer can identify a signature that captures a certain subject matter (such as amputations), generate a uniform interpolant that axiomatizes this subject matter, and then inspect it for problems. Another example is ontology re-use, where it may be more appropriate to import a uniform interpolant covering only the relevant terms than being forced to import additional terms as well.

For many logical languages, it is known that uniform interpolants do not always exist; this is true for FO, but also for various modal logics [10,30]. Positive results are known for propositional intuitionistic logic [24] and the modal μ-calculus [7]. Thus, the applicability of uniform interpolants crucially depends on the ontology language used and the language $\mathcal{L}$ in which uniform interpolants are to be axiomatized. The following simple example shows that even for $\mathcal{EL}$, uniform interpolants do not always exist unless one admits second-order expressivity in $\mathcal{L}$: let $\Sigma = \{A, r\}$ and

$$T = \{\forall x\ (A(x) \rightarrow B(x)), \forall x\ (B(x) \rightarrow \exists y(r(x,y) \wedge B(y))\}.$$

The class of Σ-reducts of T-models coincides with the models satisfying

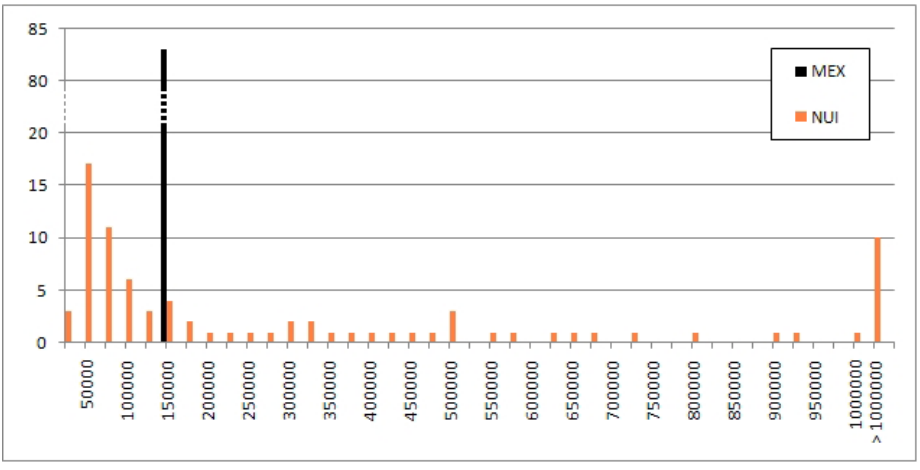
$$A(x) \rightarrow (\text{there exist } x_1, x_2, \ldots \text{ such that } r(x,x_1), r(x_1,x_2), \ldots)$$

and the first-order theory of this class of models is axiomatized by

$$\{A(x) \rightarrow \exists x_1 \cdots \exists x_n\ (r(x,x_1) \wedge \cdots \wedge r(x_{n-1},x_n)) \mid n > 0\}$$

which is not finitely axiomatizable in FO. We are thus again faced with the problem that a naive application of tools from mathematical logic is only of limited use, and a more careful analysis of the situation is required. We explore three options.

First, there are useful cases in which uniform interpolants are guaranteed to exist: observe that, although rather simple, the theory T above is not unfoldable. Indeed, it turns out [15,16] that for unfoldable $\mathcal{EL}$-theories, uniform interpolants w.r.t. $\mathcal{EL}$-implications always exist. These uniform interpolants can be of exponential size in the worst case, and so their practical relevance can only be evaluated with the help of experiments. Figure 3.2, which is taken from [16], compares the size of minimal modules based on model-conservative



extensions extracted from SNOMED CT using the prototype implementation MEX with the size of uniform interpolants computed using the prototype implementation NUI. For 83 randomly generated signatures consisting of 3000 unary and 20 binary predicates, it gives the number of modules and uniform interpolants (vertical axis) of a given size (horizontal axis), where the size is measured as as the number of symbols that occur in the module/interpolant. Thus, 10 uniform interpolants consist of more than 1 000 000 symbols. For all 83 signatures, the size of corresponding modules is between 125 000 and 150 000 symbols (which is about 3% of full SNOMED CT). Note that, in contrast to modules, the size of uniform interpolants depends a lot on the concrete signature. It is not clear yet, however, whether the very large uniform interpolants have a smaller axiomatization not found by the prototype implementation. Similar results hold for uniform interpolants formulated in languages $\mathcal{L}$ that correspond to query conservativity as discussed above. These first experiments suggest that uniform interpolants can become a useful tool for ontology engineering.

Second, even in setups where uniform interpolants are not guaranteed to exist, it might well be the case that for those ontologies that are actually used, uniform interpolants *do* usually exist. Then, it is of interest to develop algorithms that decide, given a signature Σ and an ontology T, whether a uniform interpolant exists. An example of such a result is the following: there do not always exist uniform interpolants for $\mathcal{ALC}$-theories (with uniform interpolants also formulated in $\mathcal{ALC}$), but it is decidable whether a uniform interpolant exists. Tight complexity bounds and experimental evaluation is still missing, though.

Third, one can move to a language $\mathcal{L}$ for the uniform interpolant that admits second-order expressivity. The modal μ-calculus has uniform interpolation, and so a natural option is to consider as the language $\mathcal{L}$ extensions of $\mathcal{EL}$, $\mathcal{ALC}$, and OWL-DL with fixpoint operators.

4 Discussion

We have discussed how two notions introduced in mathematical logic, conservative extensions and uniform interpolation, can be applied to ontology engineering. Many other notions from mathematical logic remain to be explored. An example is *interpretations* between logical theories, which can be regarded as a generalization of conservative extensions. In many applications of ontologies such as data integration, mappings between the symbols of ontologies are of crucial importance. Investigating the relation between the mappings used by ontology engineers and theory interpretations as known from mathematical logic could be of great interest. Another research direction is abstract model theory for ontology languages: a main problem in the field of ontology languages is a lack of logic-based criteria to classify languages and understand their place within the landscape of all potential ontology languages. Currently, only concrete languages are investigated and compared, and methods from abstract model theory might well lead to a better understanding.

References

1. Foundational Model Explorer, http://fme.biostr.washington.edu:8089/FME/index.html
2. http://www.ihtsdo.org/
3. KR-Med: Representing and sharing knowledge using SNOMED. In: CEUR Workshop Proceedings, vol. 410 (2008)
4. Andreka, H., Madarasz, J., Nemeti, I.: Logic of space-time and relativity theory. In: Handbook of Spatial logic, pp. 607–711. Springer, Heidelberg (2007)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. IJCAI. Morgan Kaufmann, San Francisco (2005)
6. Baader, F., Calvanes, D., McGuiness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, implementation and applications. Cambridge University Press, Cambridge (2003)
7. D'Agostino, G., Lenzi, G.: An axiomatization of bisimulation quantifiers via the μ-calculus. Theoret. Comput. Sci. 338 (2005)
8. Eiter, T., Wang, K.: Semantic forgetting in answer set programming. Artif. Intell. 172(14), 1644–1672 (2008)
9. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logic. In: Proc. KR, pp. 187–197 (2006)
10. Ghilardi, S., Zawadowski, M.: Undefinability of propositional quantifiers in the modal system S4. Studia Logica 55 (1995)
11. Golbeck, J., Fragaso, G., Hartel, F., Hendler, J., Parsia, B., Oberhaler, J.: The national cancer institute's thesaurus and ontology. J. of Web Semantics (2003)
12. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. Journal of Artificial Intelligence Research 31, 273–318 (2008)
13. Hofmann, M.: Proof-theoretic approach to description-logic. In: Proc. LICS, pp. 229–237 (2005)
14. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logic. In: Proc. ECAI, pp. 55–59 (2008)

15. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS, vol. 5195, pp. 259–274. Springer, Heidelberg (2008)
16. Konev, B., Walther, D., Wolter, F.: Forgetting and uniform interpolation in large-scale description logic terminologies. In: Proceedings of IJCAI (2009)
17. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in el using a relational database system. In: Proceedings of IJCAI (2009)
18. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. IJCAI, pp. 453–458 (2007)
19. Lutz, C., Wolter, F.: Conservative extensions in the lightweight description logic $\mathcal{EL}$. In: Pfenning, F. (ed.) CADE 2007. LNCS, vol. 4603, pp. 84–99. Springer, Heidelberg (2007)
20. Lutz, C., Wolter, F.: Deciding inseparability and conservative extensios in the description logic EL. Journal of Symbolic Computation (2009)
21. Mosses, P.D. (ed.): CASL Reference Manual. LNCS, vol. 2960. Springer, Heidelberg (2004)
22. I. H. T. S. D. Organisation. SNOMED CT User Guide (2008), http://www.ihtsdo.org/snomed-ct/snomed-ct-publications/
23. Patel, C., Cimino, J., Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Schonberg, E., Srinivas, K.: Matching patient records to clinical trials using ontologies. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 816–829. Springer, Heidelberg (2007)
24. Pitts, A.: On an interpretation of second-order quantification in first-order intuitionistic propositional logic. J. Symbolic Logic 57 (1992)
25. Diaconescu, J.G.R., Stefaneas, P.: Logical support for modularisation. In: Huet, G., Plotkin, G. (eds.) Logical Environments (1993)
26. Rector, A.L., Rogers, J.: Ontological and practical issues in using a description logic to represent medical concept systems: Experience from galen. In: Reasoning Web, pp. 197–231 (2006)
27. Reiter, R., Lin, F.: Forget it? In: Proceedings of AAAI Fall Symposium on Relevance, pp. 154–159 (1994)
28. Spackman, K.: Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. J. of the Amer. Med. Informatics Ass. (2000); Fall Symposium Special Issue
29. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies. LNCS, vol. 5445. Springer, Heidelberg (2009)
30. Visser, A.: Uniform interpolation and layered bisimulation. In: Gödel 1996 (Brno, 1996). Lecture Notes Logic, vol. 6. Springer, Heidelberg (1996)

Recognizability in the Simply Typed Lambda-Calculus

Sylvain Salvati

INRIA Bordeaux – Sud-Ouest,
351, cours de la Libération Bâtiment A29
33405 Talence cedex France
sylvain.salvati@labri.fr

Abstract. We define a notion of recognizable sets of simply typed λ-terms that extends the notion of recognizable sets of strings or trees. This definition is based on finite models. Using intersection types, we generalize the notions of automata for strings and trees so as to grasp recognizability for λ-terms. We then expose the closure properties of this notion and present some of its applications.

1 Introduction

Formal language theory is mainly concerned with the study of structures like strings, trees or even graphs. In this paper we try to add simply typed λ-terms to the scope of this theory. This article is a first step: the definition of recognizable sets which are a fundamental notion of formal language theory.

Languages of λ-terms appear in several research areas, but there has been really few research explicitly mentioning them and even fewer studying them. To our knowledge the first work explicitly defining a notion of language of λ-terms is that of de Groote [1]. In mathematical linguistics, the pioneering work Montague [2] shows how to connect syntax and semantics of natural language with the simply typed λ-calculus. Syntactic structures are interpreted via a homomorphism built with λ-terms. The normal forms obtained this way denote formulae of higher-order logic whose interpretation in a suitable model gives the semantics of the sentence. The set of formulae that this technique allows to generate can be seen as a language and one may wonder whether such a language can be *parsed* similarly to other languages like context-free languages. Parsing such languages results in generating sentences from their meaning representation. We have showed that this could effectively be done [3].

Still in mathematical linguistics, the type-logical tradition originating from Lambek's work [4], defines syntactic structures as proofs in some substructural logic. Several proposals have emerged in order to control the structure of those proofs such as in Moortgat's work [5] and his followers. These proofs may be represented as simply typed λ-terms and the set of syntactic structures defines a language of λ-terms.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 48–60, 2009.

Since simply typed λ-terms generalize both strings and trees, a notion of recognizable language of simply typed λ-terms should naturally extend those already defined for strings and trees. Furthermore, these languages should also be closed under the operational semantics of the λ-calculus, *i.e.* $\beta\eta$-convertibility. The easiest way to obtain such an extension is to use the algebraic characterization of recognizable sets of strings or trees which says that recognizable sets are precisely the sets that are the union of equivalence classes of a finite congruence. Generalizing this definition to sets of simply typed λ-terms consists in saying that such sets are recognizable if and only if they the set of terms that are interpreted as certain points in a finite model. But such a definition may not be useful in certain situations, this is the reason why we need a notion of automaton of λ-terms that coincides with that of recognizability. We define such automata using intersection types.

This work provides a natural framework in which several results that have appeared in the literature on simply typed λ-calculus can be related. In particular, our work shows that Urzyczyn's result on the undecidability of the emptiness problem for intersection types[6] can be seen as a corollary of Loader's result on the undecidability of λ-definability [7]. Moreover, we have showed in [3] that the singleton language can be defined with intersection types, the equivalence we establish here between recognizability in terms of finite models and in terms of automata gives an alternate proof of Statman's completeness theorem [8] (see also [9]). Furthermore, Statman [8] has showed that higher order matching is related to λ-definability. Since our notion of recognizability is related to λ-definability it gives tools with which we can study this problem.

The paper is organized as follows: section 2 gives the necessary definitions, section 3 gives the definition of recognizable sets of λ-terms. In section 4 we give an automaton-like characterization of recognizability. Section 5 gives its closure properties and section 6 shows some basic applications of the notion of recognizability for parsing and higher order matching. Finally we conclude in section 7.

2 Preliminaries

We here briefly review various notions concerning the simply typed λ-calculus and some related notions.

2.1 Simply Typed λ-Calculus

Higher Order Signatures (HOS) declare a finite number of constants by assigning them types. An HOS Σ is a triple $(\mathcal{A}, \mathcal{C}, \tau)$, where $\mathcal{A}$ is a finite set of *atomic types*, from which the set of *complex types*, $\mathcal{T}_\Sigma$, is built using the binary infix operator $\rightarrow$, $\mathcal{C}$ is a finite set of constants and τ is a function from $\mathcal{C}$ to $\mathcal{T}_\Sigma$. As usual, we will consider that $\rightarrow$ associates to the right and write $\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha$ for the type $(\alpha_1 \rightarrow (\cdots \rightarrow (\alpha_n \rightarrow \alpha) \cdots))$. The *order* of a type α, $ord(\alpha)$, is 1 when α is atomic and $\max(ord(\alpha_1) + 1, ord(\alpha_2))$ when $\alpha = \alpha_1 \rightarrow \alpha_2$. By extension,

the order of a HOS is the maximal order of type it associates to a constant. We suppose that we are given an infinite countable set of λ-variables V. We use types *à la Church* and variables explicitly carry their types. So we will write x^α, y^α or z^α (possibly with indices) the elements of $\mathcal{V} \times \{\alpha\}$, the variables of type α. A HOS Σ defines a set of simply typed λ-terms Λ_Σ. This set is the union of the sets of the family $(\Lambda^\alpha_\Sigma)_{\alpha \in \mathcal{T}_\Sigma}$ defined as the smallest sets verifying:

1. $x^\alpha \in \Lambda^\alpha_\Sigma$,
2. for $c \in \mathcal{C}$, $c \in \Lambda^{\tau(c)}_\Sigma$,
3. if $M_1 \in \Lambda^{\alpha\to\beta}_\Sigma$ and $M_2 \in \Lambda^\alpha_\Sigma$ then $(M_1 M_2) \in \Lambda^\beta_\Sigma$,
4. if $M \in \Lambda^\beta$ then $\lambda x^\alpha.M \in \Lambda^{\alpha\to\beta}_\Sigma$.

We take for granted the notions of free variables, closed terms, substitution, the various notions of conversions and reductions associated to the λ-calculus, the notions of normal form, of η-long form (or long form) and the notion of linearity. We write $\Lambda^\alpha_{\Sigma,W}$ to designates the set of terms of type α built on Σ whose set of free variables is included in W.

2.2 Trees and Strings as λ-Terms

A HOS is said to be a tree-HOS when it is a second order HOS and that it uses only one type namely o. We write $o^n \to o$ in the place of $\underbrace{o \to \cdots \to o}_{n} \to o$ and say that a constant of type $o^n \to o$ has *arity* n. It is easy to see that every freely and finitely generated sets of ranked trees can be seen as a the closed normal terms of type o built on a tree-HOS.

A HOS is said to be a string-HOS when it is a tree-HOS whose constants all have arity 1. Strings are represented as closed terms of type $o \to o$ and the string $c_1 \dots c_n$ is represented by the term $\lambda x^o.c_1(\dots(c_n x^o)\dots)$ denoted by $/c_1 \dots c_n/$. The empty string is $\lambda x^o.x^o$ and the concatenation operation can be represented by function composition $\lambda x^o.s_1(s_2\, x^o)$ (*c.f.* [1]).

2.3 Homomorphisms

A *homomorphism* between the signatures Σ_1 and Σ_2 is a pair (g, h) such that g maps $\mathcal{T}_{\Sigma_1}$ to $\mathcal{T}_{\Sigma_2}$, h maps Λ_{Σ_1} to Λ_{Σ_2} and verify the following properties:

1. $g(\alpha \to \beta) = g(\alpha) \to g(\beta)$,
2. $h(x^\alpha) = x^{g(\alpha)}$,
3. $h(c)$ is a closed term of $\Lambda^{g(\tau(c))}_{\Sigma_2}$,
4. $h(M_1 M_2) = h(M_1)h(M_2)$ and
5. $h(\lambda x^\beta.M) = \lambda x^{g(\beta)}.h(M)$.

A homomorphism is said to be *linear* whenever constants are mapped to linear terms. We write $\mathcal{H}(\alpha)$ and $\mathcal{H}(M)$ respectively instead of $g(\alpha)$ and of $h(M)$ for a given homomorphism $\mathcal{H} = (g, h)$. Note that if $\mathcal{H}$ is a homomorphism from Σ_1 to Σ_2 and $M \in \Lambda^\alpha_{\Sigma_1}$ then $\mathcal{H}(M) \in \Lambda^{\mathcal{H}(\alpha)}_{\Sigma_2}$.

The order of a homomorphism $\mathcal{H}$ is the maximal order of the type it associates to an atomic type. The usual notion of tree-homomorphism (*resp.* string homomorphism) is a first order homomorphism (in our sense) between tree-signatures (*resp.* string-signatures). A first order homomorphism between Σ_1 and Σ_2 that maps constants of Σ_1 to constants of Σ_2 is called a *relabeling*.

2.4 Models

Given a signature Σ, a *full model* of Σ is a pair $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ where:

1. $(\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}$ is a family of sets verifying:
   (a) for all $\alpha, \beta \in \mathcal{T}_\Sigma$, $\mathcal{M}^{\alpha\to\beta} = \mathcal{M}^{\alpha\mathcal{M}^\beta}$,
   (b) the sets $\mathcal{M}^\alpha$ such that α is atomic are pairwise disjoint.
2. ρ is a function from $\mathcal{C}$ to $\mathcal{M}^\alpha$ so that $\alpha = \rho(c)$.

A full model, $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$, of Σ is said *finite* when for all $\alpha \in \mathcal{T}_\Sigma$, $\mathcal{M}^\alpha$ is a finite set. Remark that $\mathbb{M}$ is finite if and only if for all atomic types α, $\mathcal{M}^\alpha$ is finite.

Given $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a full model of Σ, the terms of Λ^α_Σ are interpreted as elements of $\mathcal{M}^\alpha$. This interpretation necessitates the definition of *variable assignments* which are partial functions that associate elements of $\mathcal{M}^\alpha$ to variables like x^α. A variable assignment is said *finite* when its domain is finite. Given a variable assignment ν, a variable x^α and $m \in \mathcal{M}^\alpha$ we define $\nu[x^\alpha \leftarrow m]$ to be the variable assignment verifying:

$$\nu[x^\alpha \leftarrow m](y^\beta) \begin{cases} m \text{ if } y^\beta = x^\alpha \\ \nu(y^\beta) \text{ otherwise} \end{cases}$$

Given a full model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a variable assignment ν, the interpretation of the elements of Λ_Σ (whose sets of free variables are included in the domain of definition of ν) in $\mathbb{M}$ is inductively defined as follows:

1. $[\![x^\alpha]\!]^{\mathbb{M}}_\nu = \nu(x^\alpha)$
2. $[\![c]\!]^{\mathbb{M}}_\nu = \rho(c)$
3. $[\![M_1 M_2]\!]^{\mathbb{M}}_\nu = [\![M_1]\!]^{\mathbb{M}}_\nu([\![M_2]\!]^{\mathbb{M}}_\nu)$
4. $[\![\lambda x^\alpha.M]\!]^{\mathbb{M}}_\nu$ is the function which maps $m \in \mathcal{M}^\alpha$ to $[\![M]\!]^{\mathbb{M}}_{\nu[x^\alpha\leftarrow m]}$.

It is well-known that the semantics of λ-terms is invariant modulo $\beta\eta$-reduction.

3 Recognizable Sets of λ-Terms

We wish to extend the notion of recognizability that already exists for strings and trees to λ-terms. An abstract way of defining recognizability for strings and trees is to use Myhill-Nerode theorem [10], [11], that describes it in terms of congruence of finite index over strings or trees which is equivalent to describing it in terms of finite algebra for trees or finite semigroups for strings. This approach has been successfully extended in the seminal paper [12] to any abstract algebra. We shall follow this line of work in order to define recognizability for the simply typed λ-calculus. Since the finite full models form the functional closure of finite algebra, we use them so as to extend recognizability to λ-terms.

Definition 1. *Given a HOS Σ and $\alpha \in \mathcal{T}_\Sigma$ a set $\mathcal{R}$ included in Λ^α_Σ is said to be recognizable iff there is a finite and full model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ a finite variable assignment ν and a subset $\mathcal{P}$ of $\mathcal{M}^\alpha$ such that: $\mathcal{R} = \{M | [\![M]\!]^{\mathbb{M}}_\nu \in \mathcal{P}\}$.*

Note that in this definition when ν is chosen to be the empty assignment function then the set $\mathcal{R}$ only contains closed terms. In particular, when Σ is a tree (*resp.* string) signature, and that α is the atomic type o (*resp.* the type $o \to o$) then the set of closed λ-terms that are recognizable correspond exactly to set of recognizable trees (*resp.* strings).

The result by Loader [7] shows that in general it is undecidable to check whether a recognizable set is empty. But as soon as the finite and full model and the assignment function are given we can check whether a term is in the set. In what follows we give a mechanical way (corresponding to automata for trees or strings) to define recognizable sets and check whether a certain term belongs to that set.

A classical and simple example of recognizable set of trees being the set of true boolean formulae, we exemplify the notion of recognizability for λ-terms with the set of true Quantified Boolean Formulae (QBF). For this it suffices to use a HOS B whose constants are: $\wedge : b^2 \to b$, $\vee : b^2 \to b$, $\neg : b \to b$, $\mathbf{1} : b$, $\mathbf{0} : b$, $\forall : (b \to b) \to b$ and $\exists : (b \to b) \to b$. We use a finite model $\mathbb{B} = ((\mathcal{B}^\alpha)_{\alpha\in\mathcal{T}_B}, \rho)$ such that $\mathcal{B}^b = \{0; 1\}$ and ρ associates the obvious functions to the constants of B. Then the set of terms representing a true QBF is the set of closed λ-terms of B^b which are interpreted as 1 in $\mathbb{B}$ and therefore this set is recognizable.

4 Automata Characterizing Recognizable Sets

We here generalize the notion of automata for trees and strings in order to obtain a mechanical definition of recognizability for λ-terms. Our notion of automaton is based on the notion of *Higher Order Intersection Signature (HOIS)* introduced in [3] which, in turn, is based on intersection types [13]. A HOIS is a tuple $\Pi = (\Sigma, I, \iota, \chi)$ where Σ is a HOS, I is a finite set of *atomic intersection types*, ι is a function from I to the atomic types of Σ, χ is a function that associates to every element of $\mathcal{C}$ a subset of $\cap^{\tau(c)}_\Pi$ where $(\cap^\alpha_\Pi)_{\alpha\in\mathcal{T}_\Sigma}$ is the smallest family verifying:

1. for α and atomic type $\cap^\alpha_\Pi = \iota^{-1}(\alpha)$,
2. $\cap^{\alpha\to\beta}_\Pi = 2^{\cap^\alpha_\Pi} \times \{\alpha\} \times \cap^\beta_\Pi$ (we write 2^P, the powerset of the set P)

A trivial induction on the structure of α shows that for each type α, the set $\cap^\alpha_\Pi$ is finite.

We now define the type system that allows to associate types to λ-terms. Given a HOIS $\Pi = (\Sigma, I, \iota, \chi)$, a *$\Pi$-typing environment* (or simply, *typing environment*, when there is no ambiguity) , is a partial mapping from typed variables to $2^{\cap_\Pi}$ whose domain is finite and such that $\Gamma(x^\alpha)$, when it is defined, is included in $\cap^\alpha_\Pi$. We write $\overline{\Gamma}$ to denote the domain of Γ. *Judgements over Π* are objects of the form $\Gamma \vdash_\Pi M : p$ where Γ is a typing environment, M is an element of

Λ_Σ and p is an element of $\cap_\Pi$. Judgements are derived by using the following inference system:

$$\frac{p \in \Gamma(x^\alpha)}{\Gamma \vdash_\Pi x^\alpha : p}\ \textsc{Axiom} \qquad \frac{p \in \chi(c)}{\Gamma \vdash_\Pi c : p}\ \textsc{Const} \qquad \frac{\Gamma \vdash_\Pi M : p \quad p \sqsubseteq^\alpha_\Pi q}{\Gamma \vdash_\Pi M : q}\ \textsc{Sub}$$

$$\frac{\Gamma, x^\alpha : S \vdash_\Pi M : p}{\Gamma \vdash_\Pi \lambda x^\alpha.M : (S, \alpha, p)}\ \textsc{Abst}$$

$$\frac{\Gamma \vdash_\Pi M : (S, \alpha, p) \quad N \in \Lambda^\alpha_{\Sigma,\overline{\Gamma}} \quad \forall q \in S.\ \Gamma \vdash_\Pi N : q}{\Gamma \vdash (MN) : p}\ \textsc{App}$$

Where the relation $\sqsubseteq^\alpha_\Pi$ is defined as follows:

$$\frac{i \in \iota(\alpha)}{i \sqsubseteq^\iota_\Pi i} \qquad \frac{T \subseteq \cap^\alpha_\Pi \quad \forall p \in S. \exists q \in T. q \sqsubseteq^\alpha_\Pi p}{T \trianglelefteq^\alpha_\Pi S} \qquad \frac{S \trianglelefteq^\alpha_\Pi T \quad q \sqsubseteq^\beta_\Pi p}{(T, \alpha, q) \sqsubseteq^{\alpha\to\beta}_\Pi (S, \alpha, p)}$$

Notice that the rule App, has two premises, concerning N. The reason of being of the premise $N \in \Lambda^\alpha_{\Sigma,\overline{\Gamma}}$ is that when M has an intersection type of the form $(\emptyset, \alpha, p)$, the premise $\forall q \in S.\ \Gamma \vdash_\Pi N : q$ is trivially true and without such a premise we would derive judgments on terms which would not be simply typed.

We will use the notation $\Gamma \vdash_\Pi M : S$ where S is a subset of $\cap^\alpha_\Pi$ to denote the all the judgements of the form $\Gamma \vdash_\Pi M : p$ where p in S. In the same spirit, given S and T that are respectively subsets of $\cap^\alpha_\Pi$ and of $\cap^\beta_\Pi$, we write (S, α, T) to denote the subset of $\cap^{\alpha\to\beta}_\Pi$, $\{(S, \alpha, p) | p \in T\}$.

We now give the principal properties of that system. The proofs of those properties can be found in [3].

Theorem 1 (subject reduction). *If $\Gamma \vdash_\Pi M : p$ is derivable and $M \twoheadrightarrow_{\beta\eta} N$ then $\Gamma \vdash_\Pi N : p$ is derivable.*

Notice that this Theorem would only hold for β-reduction without the use of the rule Sub.

Theorem 2 (subject expansion). *If $M \in \Lambda^\alpha_\Sigma$, $M \twoheadrightarrow_{\beta\eta} N$ and $\Gamma \vdash_\Pi N : p$ is derivable then $\Gamma \vdash_\Pi M : p$ is derivable.*

Theorem 3 (Singleton). *Given $M \in \Lambda^\alpha_\Sigma$, there is Π, Γ and $S \subseteq \cap^\alpha_\Pi$ such that given $N \in \Lambda^\alpha_\Sigma$, $\Gamma \vdash_\Pi N : S$ is derivable if and only if $M =_{\beta\eta} N$.*

This Singleton Theorem, requires some comments. We proved it referring to coherence theorems such as the one proved in [14]. It is also related to a Theorem proved by Statman [8], [9], since we will see that HOIS and finite full models can be represented one in the other.

Since, with intersection type we can represent graphs of functions, the set of λ-terms that are interpreted as a certain element in a finite full model are exactly the λ-terms that verify a certain judgement.

Theorem 4. *Given a HOS Σ, a finite model of Σ, $\mathbb{M} = (\mathcal{M}^\alpha, \rho)$, a finite variable assignment ν over $\mathbb{M}$ and an element e of $\mathcal{M}^\alpha$ then there is a HOIS over Σ, Π, a typing environment Γ and a subset S of $\cap_\Pi^\alpha$ such that for every λ-term M the two following properties are equivalent:*

1. $[\![M]\!]_\nu^{\mathbb{M}} = e$
2. $\Gamma \vdash_\Pi M : S$

A consequence of the previous theorem is that we can obtain the undecidability result by [6] about the emptiness of intersection type as a corollary of the undecidability of λ-decidability [7].

We now prove the converse of the previous theorem, that is, typability properties in HOIS can be seen as properties of interpretations in finite full models. The principle of this proof is to interpret intersection types as functions operating over intersection types.

We define the operator **app** which maps, for all α and β, $2^{\cap_\Pi^{\alpha\to\beta}} \times 2^{\cap_\Pi^\alpha}$ to $2^{\cap_\Pi^\beta}$. It is defined by: $\mathbf{app}(S,T) = \{p | \exists (Q,\alpha,p) \in S. T \trianglelefteq Q\}$

The finite model in which we will interpret intersection types built over Π is $\mathbb{M}_\Pi = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{I}_\mathcal{A}}, \rho)$ where for α atomic we let $\mathcal{M}^\alpha$ be $2^{\iota^{-1}(\alpha)}$. The definition of ρ requires an auxiliary function $\mathcal{F}^\alpha$ that sends subsets of $\cap_\Pi^\alpha$ to subsets of $\mathcal{M}^\alpha$ and that verifies:

1. for α atomic and S included in $\cap_\Pi^\alpha$ we let $\mathcal{F}^\alpha(S) = \{T \subseteq \cap_\Pi^\alpha | S \subseteq T\}$,
2. for S included in $\cap_\Pi^{\alpha\to\beta}$ we let

$$\mathcal{F}^{\alpha\to\beta}(S) = \{h \in \mathcal{M}^{\alpha\to\beta} | \forall T \subseteq \cap_\Pi^\alpha . \forall g \in \mathcal{F}^\alpha(T). h(g) \in \mathcal{F}^\beta(\mathbf{app}(S,T))\}$$

It is easy to verify that for every S included in $\cap_\Pi^\alpha$, the set $\mathcal{F}^\alpha(S)$ is not empty. We choose $\rho(c)$ as an element of $\mathcal{F}^{\tau(c)}(\chi(c))$. We then have the following theorem.

Theorem 5. *Given a HOS Σ, a HOIS Π over Σ, Γ and S a subset of $\cap_\Pi^\alpha$, we set $\nu(x^\alpha)$ to be an element of $\mathcal{F}_\alpha(\Gamma(x^\alpha))$, then the two following properties are equivalent:*

1. $\Gamma \vdash_\Pi M : S$
2. $[\![M]\!]_\nu^{\mathbb{M}_\Pi}$ *belongs to* $\mathcal{F}_\alpha(S)$

The Theorems 4 and 5 relate finite models and typability in HOIS. This leads us to the definition of a generalized notion of automaton, *typing-automata*.

Definition 2. *A* typing-automaton*, $\mathcal{A}$, over a HOS Σ is a tuple $(\alpha, \Pi, \Gamma, \{S_1; \ldots S_n\})$ where: $\alpha \in \mathcal{T}_\Sigma$, Π is a HOIS over Σ, Γ is a Π-typing environment, for all i in $\{1; \ldots; n\}$, S_i is a subset of $\cap_\Pi^\alpha$. The language defined by $\mathcal{A}$ is*

$$L(\mathcal{A}) = \{M | \exists i \in \mathbb{N}.\ \Gamma \vdash_\Pi M : S_i\}$$

Using Theorems 4 and 5 we get:

Theorem 6. *A language of λ-terms L is recognizable if and only if there is a typing-automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.*

5 Closure Properties

5.1 Boolean Closure

In this section we shall quickly outline how to construct of typing-automata for the boolean closure properties of recognizable sets of λ-terms. Interestingly these constructions can be seen as generalizations of the usual constructions that are used for tree/string-automata. For example, concerning the intersection of two recognizable languages, we can construct the product of two typing-automata. We first start by defining the product of two HOIS.

Definition 3. *Given* $\Pi_1 = (\Sigma, I_1, \iota_1, \chi_1)$ *and* $\Pi_2 = (\Sigma, I_2, \iota_2, \chi_2)$ *we define the HOIS* $\Pi_1 \otimes \Pi_2$ *to be* (Σ, I, ι, χ) *where:*

1. I *is a subset of* $I_1 \times I_2$ *which is equal to* $\{(p_1, p_2) | \iota_1(p_1) = \iota_2(p_2)\}$,
2. $\iota((p_1, p_2)) = \iota_1(p_1)$, *note that by definition of* I, $\iota((p_1, p_2)) = \iota_2(p_2)$,
3. $\chi(c) = \{p_1 \otimes p_2 | p_1 \in \chi_1(c)$ *and* $p_2 \in \chi_2(c)\}$.

where given p_1 *in* $\cap^{\alpha}_{\Pi_1}$ *and* p_2 *in* $\cap^{\alpha}_{\Pi_2}$ *we define* $p_1 \otimes p_2$ *by:*

1. *if* α *is atomic then* $p_1 \otimes p_2 = (p_1, p_2)$
2. *if* $\alpha = \alpha_1 \to \alpha_2$ *then* $p_1 = (S_1, \alpha_1, q_1)$ *and* $p_2 = (S_2, \alpha_2, q_2)$ *and* $p_1 \otimes p_2 = (S_1 \otimes S_2, \alpha_1, q_1 \otimes q_2)$ *where* $S_1 \otimes S_2 = \{r_1 \otimes r_2 | r_1 \in S_1$ *and* $r_2 \in S_2\}$

If we define the product of two typing environment Γ and Δ to be $\Gamma \otimes \Delta$ such that $\Gamma \otimes \Delta(x) = \Gamma(x) \otimes \Delta(x)$, we can prove the following property:

Theorem 7. *The judgements* $\Gamma \vdash_{\Pi_1} M : P$ *and* $\Delta \vdash_{\Pi_2} M : Q$ *are derivable if and only if the judgement* $\Gamma \otimes \Delta \vdash_{\Pi_1 \otimes \Pi_2} M : P \otimes Q$ *is derivable.*

This allows us to define the product $\mathcal{A} \otimes \mathcal{B}$ of two typing-automata $\mathcal{A}$ and $\mathcal{B}$.

Definition 4. *Given two typing-automata over some HOS* Σ, $\mathcal{A} = (\alpha, \Pi_1, \Gamma, T_1)$ *and* $\mathcal{B} = (\alpha, \Pi_2, \Delta, T_2)$, *we let* $\mathcal{A} \otimes \mathcal{B}$ *be* $(\alpha, \Pi_1 \otimes \Pi_2, \Gamma \otimes \Delta, T_1 \otimes T_2)$ *where* $T_1 \otimes T_2$ *is the set* $\{S_1 \otimes S_2 | S_1 \in T_1$ *and* $S_2 \in T_2\}$.

Theorem 8. *Given two typing automata of the same type over some HOS* Σ, $\mathcal{A}$ *and* $\mathcal{B}$ *we have* $L(\mathcal{A} \otimes \mathcal{B}) = L(\mathcal{A}) \cap L(\mathcal{B})$.

The closure under complement of the class of recognizable sets of λ-terms, is a direct consequence of its definition in terms of finite models. Interestingly, if one wishes to construct the typing-automaton recognizing the complementary language of a given typing-automaton, then one would use the construction that serves in Theorem 5 which on a tree or string automaton would corresponds to *determinization*. This induces a notion of deterministic typing-automata that grasps the notion of recognizability, and corresponding to the fact that intersection types correspond to partial function over the finite model generated by the atomic intersection types.

5.2 Homomorphisms

It is well-known that recognizable sets of strings are closed under arbitrary homomorphisms while recognizable sets of trees are closed under linear homomorphisms. We will see that recognizable sets of λ-terms are not even closed under relabeling. This has the consequence, that Monadic Second Order Logic (MSO) over the structure of normal λ-terms is not grasped by our notion of recognizability, since relabelings allow to represent set quantification. On the other hand, alike strings and trees, recognizable sets of λ-terms are closed under arbitrary inverse homomorphisms.

We now turn to show that recognizable sets of λ-terms are not closed under relabeling. In order to show this we use the following signature $\Sigma = \{\forall : (b \to b) \to b, \wedge : b \to b \to b, \vee : b \to b \to b, \neg : b \to b, \lhd : b \to b \to b, \rhd : b \to b \to b\}$. Since terms built on Σ are usual boolean expressions, we shall use the standard notation for those expressions instead of the λ-term notation. Thus we shall write $\forall x.t$, $t_1 \wedge t_2$ and $t_1 \vee t_2$ instead of $\forall(\lambda x.t)$, $\wedge t_1\, t_2$ and $\vee t_1\, t_2$. The terms built on Σ are interpreted in a finite model $\mathbb{B} = ((\mathcal{B}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ where $\mathcal{B}^b = \{0; 1\}$ and ρ interprets the usual boolean connectives and quantifiers ($\wedge$, $\vee$, $\neg$ and $\forall$) with their usual truth tables and ρ interprets the connectives $\lhd$ and $\rhd$ as the functions such that $\rho(\lhd)xy = x$ and $\rho(\rhd)xy = y$. By definition the set $\mathcal{T}$ of closed terms whose semantic interpretation in $\mathbb{B}$ is 1 is recognizable.

We use a relabeling $\mathcal{H}$ which maps the terms built on Σ to terms built on $\Sigma' = \{\forall : (b \to b) \to b, \wedge : b \to b \to b, \vee : b \to b \to b, \neg : b \to b, \bowtie\colon b \to b \to b\}$ so the constants $\forall$, $\wedge$, $\vee$ and $\neg$ are mapped to themselves by $\mathcal{H}$ and $\lhd$ and $\rhd$ are both mapped to $\bowtie$.

We let $\Leftrightarrow$ be the λ-term $\lambda xy.(x \wedge y) \vee (\neg x \wedge \neg y)$; as for the other connective, we adopt an infix notation, *i.e.* we shall write $t_1 \Leftrightarrow t_2$ instead of $\Leftrightarrow t_1\, t_2$.

As the connective $\lhd$ (*resp.* $\rhd$) takes the value of its left (*resp.* right) argument, if f and g are terms whose free variables are $x_1, \ldots, x_n$, then we have the following identities $[\![\forall x_1 \ldots \forall x_n.(\lhd f\, g) \Leftrightarrow f]\!]^{\mathbb{B}} = 1$ and $[\![\forall x_1 \ldots \forall x_n.(\rhd f\, g) \Leftrightarrow g]\!]^{\mathbb{B}} = 1$.

The closed term $\lambda x_1^b \ldots x_n^b.t$ built on Σ can be interpreted as a function from $\{0; 1\}^n$ to $\{0; 1\}$ (modulo curryfication) in $\mathbb{B}$, *i.e.* an n-ary boolean function. For a given n there are 2^{n+1} such functions and we know that for each such function f we can build, using only $\wedge$, $\vee$ and $\neg$, a term $\widetilde{f}$ such that $[\![\lambda x_1 \ldots x_n.\widetilde{f}]\!]^{\mathbb{B}} = f$. Given $\mathcal{F} = \{f_1; \ldots; f_p\}$ a set of such functions, we write $[\mathcal{F}]$ the term $\bowtie \widetilde{f}_1(\bowtie \widetilde{f}_2(\ldots(\bowtie \widetilde{f_{p-1}}\widetilde{f_p})\ldots))$. Remark that for all i in $\{1; \ldots; p\}$, there is H_i such that $\mathcal{H}(H_i) = [\mathcal{F}]$ and $[\![\forall(x_1 \ldots \forall(x_n.H_i \Leftrightarrow \widetilde{f}_i)\ldots)]\!]^{\mathbb{B}} = 1$ and thus $\forall(x_1 \ldots \forall(x_n.[\mathcal{F}] \Leftrightarrow \widetilde{f}_i)$ is in $\mathcal{H}(\mathcal{T})$. Furthermore for every H such that $\mathcal{H}(H) = \mathcal{F}$ there is i in $\{1; \ldots; p\}$ such that $[\![\forall(x_1 \ldots \forall(x_n.H \Leftrightarrow \widetilde{f}_i)\ldots)]\!]^{\mathbb{B}} = 1$. If we suppose that $\mathcal{H}(\mathcal{T})$ is recognizable, then there is a finite model $\mathbb{M} = ((\mathcal{M}^\alpha)_{\alpha\in\mathcal{T}_\Sigma}, \rho)$ and a subset $\mathcal{N}$ of $\mathcal{M}^b$ such that the closed terms M are in $\mathcal{H}(\mathcal{T})$ if and only if $[\![M]\!]^{\mathbb{M}} \in \mathcal{N}$; we assume that $\mathcal{M}^b$ contains q elements. Each closed term $\lambda x_1^b \ldots x_n^b.M$ built on Σ' is interpreted in $\mathbb{M}$ as a function from $\{1; \ldots; q\}^n$ to $\{1; \ldots; q\}$ (modulo curryfication). We are going to show that for every sets of n-ary boolean functions $\mathcal{F}$ and $\mathcal{G}$, it is necessary that $[\![\lambda x_1^b \ldots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ and

$[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$ are different when $\mathcal{F}$ and $\mathcal{G}$ are different. Indeed, if $\mathcal{F}$ and $\mathcal{G}$ are different, we can assume without loss of generality that $\mathcal{F}$ is not empty, and then there is a boolean function f which is in $\mathcal{F}$ and which is not in $\mathcal{G}$. Since there is H such that $\mathcal{H}(H) = [\mathcal{F}]$ and $[\![\forall x_1. \dots \forall x_n.H \Leftrightarrow \widetilde{f}]\!]^{\mathbb{B}} = 1$, then $\forall x_1. \dots \forall x_n.[\mathcal{F}] \Leftrightarrow \widetilde{f}$ is in $\mathcal{H}(\mathcal{T})$. But for an n-ary boolean g, there is an H' such that $\mathcal{H}(H') = [\mathcal{G}]$ and $[\![\forall x_1. \dots \forall x_n.H' \Leftrightarrow \widetilde{g}]\!]^{\mathbb{B}} = 1$ iff g is in $\mathcal{G}$. Thus the term $\forall x_1. \dots \forall x_n.[G] \Leftrightarrow \widetilde{f}$ is not in $\mathcal{H}(\mathcal{T})$ and $[\![\lambda x_1^b \dots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ is different from $[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$. But there are $2^{2^{n+1}}$ sets of n-ary boolean functions while there are q^{n+1} functions from $\{1; \dots; q\}^n$ to $\{1; \dots; q\}$ and thus for n sufficiently big, it is not possible to verify that $[\![\lambda x_1^b \dots x_n^b.[\mathcal{F}]]\!]^{\mathbb{M}}$ and $[\![\lambda x_1^b \dots x_n^b.[\mathcal{G}]]\!]^{\mathbb{M}}$ are different when $\mathcal{F}$ and $\mathcal{G}$ are different. Therefore, $\mathcal{H}(\mathcal{T})$ is not a recognizable set. This implies that the class of recognizable sets of λ-terms is not closed under relabeling.

While there seems to be no interesting class of homomorphisms under which our notion of recognizability is closed, we can show that recognizable sets of λ-terms are closed under inverse homomorphism.

Theorem 9. *Given Σ_1, Σ_2 two HOS and $\mathcal{H}$ a homomorphism between Σ_1 and Σ_2, if R is a recognizable set of Σ_2 then $\mathcal{H}^{-1}(R) \cap \Lambda^{\alpha}_{\Sigma,V}$ is also recognizable.*

Recognizable sets contain only λ-terms of a given type and there is no reason why $\mathcal{H}^{-1}(R)$ is a set containing terms having all the same type. So intersecting $\mathcal{H}^{-1}(R)$ with set set of the form $\Lambda^{\alpha}_{\Sigma,V}$ is necessary.

6 Some Applications of Recognizability

We here quickly review some direct applications of the notion of recognizability in the simply typed λ-calculus.

6.1 Parsing

Theorem 9 gives a very simple definition of parsing for many formalisms. Indeed in formalisms, such as Context Free Grammars, Tree Adjoining Grammars, Multiple Context Free Grammars, Parallel Multiple Context Free Grammars *etc...* can be seen as the interpretation of trees via homomorphism (see [15]). Thus these grammars can be seen a 4-tuple $(\Sigma_1, \Sigma_2, \mathcal{H}, S)$ where Σ_1 is a multi-sorted tree signature, Σ_2 is a string signature, $\mathcal{H}$ is a homomorphism from Σ_1 to Σ_2 and S is the type of the trees that are considered as analyses. Thus if we want to parse a word w we try to find the set $\{M \in \Lambda^S_{\Sigma_1} | M \text{ is closed and } \mathcal{H}(M) =_{\beta\eta} w\}$ which is actually $\mathcal{H}^{-1}(\{w\})$. But we know from Theorem 3 that $\{w\}$ is a recognizable set and thus parsing amounts to compute the inverse homomorphic image of a recognizable set. This gives a new proof of the theorem of [16] which proves that the set of parse trees of a sentence in a context free grammars is a recognizable set, and it furthermore generalizes the result to a wide family of formalisms. Moreover, this view on parsing also applies to grammars generating tree or λ-terms, it also shows that parsing a structure is similar to parsing

recognizable sets. Parsing recognizable sets instead of singleton structures has the advantage that it allows to parse ambiguous inputs, such as noisy phonetic transcriptions, or ambiguous tagging of sentences...

6.2 Higher Order Matching

The γ-higher-order matching problem (γ-HOM), with $\gamma \in \{\beta; \beta\eta\}$, consists in solving an equation of the form $M \stackrel{?}{=}_\gamma N$ where N is a closed term. A solution of such an equation is a substitution σ such that $M.\sigma =_\gamma N$. Using the extraction Lemma of [3], and Theorem 3, it is easy to see that the solutions of $\beta\eta$-HOM form finite unions of cartesian products of recognizable sets. Observing this, allows us to obtain in an alternative way the relation between λ-definability and $\beta\eta$-HOM showed in [8]. Furthermore, we can easily obtain the result that $\beta\eta$-HOM is decidable (see [17]) when the terms in a solution are *arity bounded*, *i.e.* under the constraint that the number of variables that can be free in a subterm is bounded by some number k. Indeed, because of the subformula property and the bound on the number of free variables, arity-bounded terms of a given type can all be represented with finitely many combinators; this means that we can represent those terms in a tree-HOS Σ and recover them with a homomorphism $\mathcal{H}$. Thus, the set S of terms that are solution of arity bounded $\beta\eta$-HOM can be effectively represented as a recognizable set of trees, namely $\mathcal{H}^{-1}(S)$, the emptiness of recognizable sets of trees being decidable this gives the decidability of arity bounded $\beta\eta$-HOM. In particular, this leads to the decidability of arity bounded $\beta\eta$-HOM. Since arity-bounded $\beta\eta$-HOM is more general than 3^{rd} and 4^{th} order $\beta\eta$-HOM [17], this technique sheds some light on the results obtained by [18] that relate the solutions of these special cases to tree automata. Contrary to most approach to HOM, the one we use is completely direct, we do not need to transform the problem within a set of interpolation equations.

β-HOM [19] is undecidable while $\beta\eta$-HOM seems to be decidable [20]. But there is no satisfying explanation on the difference between β-HOM and $\beta\eta$-HOM so as to account satisfactorily of that difference. But as we have seen, intersection types make a discrimination between β-reduction and $\beta\eta$-reduction with the rule Sub, without which the subject reduction Theorem does not hold for $\beta\eta$-reduction. Thus intersection types seem to be a good tool to investigate this problem.

7 Conclusion and Future Work

We have defined a notion of recognizability for the λ-calculus that naturally extends recognizability for trees or strings. We have exhibited the closure properties of this notion and showed how it could be exploited to understand parsing of the higher order matching problem. Contrary to strings and trees where recognizability comes with three kinds of characterization, a mechanical one (automata), an algebraic one and a logical one (Monadic Second Order Logic, MSOL), here our notion only comes with a mechanical and an algebraic characterization. It

seems difficult to come up with a logical characterization since this notion is not closed under relabeling. And closure under relabeling is central to represent quantification in MSOL. As we wish to use this notion of recognizability so as to describe particular sets of λ-terms, it would be nice to obtain a connection with some logic. First-order logic would be a first step. A more general question would be whether there is a logic that exactly corresponds to this notion of recognizability.

Another question is to characterize the restrictions under which the emptiness of recognizable sets is decidable. Theorem 9 gives a positive answer when the terms are bound to be generated with a finite set of combinators since it reduces this emptiness problem to the emptiness problem of some recognizable set of trees. But we do not know whether there are weaker constraints for which this holds. When we look at the situation for graphs, there is no class of graphs [21] which can be generated only with infinitely many combinators (this means that the class of graphs has an infinite treewidth) for which this emptiness problem is decidable. Thus, this question can be related to the definition of a suitable notion for normal λ-terms that would be similar to treewidth for graphs.

Finally we hope that the notion of recognizability for λ-terms can be of interest in the study of trees generated by higher-order programming schemes. It has been showed that those trees had a decidable MSO theory [22]. It is likely that intersection types should be more adapted to conduct this proof, and yield to new techniques.

References

1. de Groote, P.: Towards abstract categorial grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference, pp. 148–155 (2001)
2. Montague, R.: Formal Philosophy: Selected Papers of Richard Montague. Yale University Press, New Haven (1974)
3. Salvati, S.: On the membership problem for Non-linear Abstract Categorial Grammars. In: Muskens, R. (ed.) Proceedings of the Workshop on New Directions in Type-theoretic Grammars (NDTTG 2007), Dublin, Ireland, Foundation of Logic, Language and Information (FoLLI), August 2007, pp. 43–50 (2007)
4. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65, 154–170 (1958)
5. Moortgat, M.: Categorial Investigations: Logical & Linguistic Aspects of the Lambek Calculus. Foris Pubns USA (1988)
6. Urzyczyn, P.: The emptiness problem for intersection types. J. Symb. Log. 64(3), 1195–1215 (1999)
7. Loader, R.: The undecidability of λ-definability. In: Anderson, C.A., Zeleny, M. (eds.) Logic, Meaning and Computation: Essays in memory of Alonzo Church, pp. 331–342. Kluwer, Dordrecht (2001)
8. Statman, R.: Completeness, invariance and λ-definability. Journal of Symbolic Logic 47(1), 17–26 (1982)
9. Statman, R., Dowek, G.: On statman's finite completeness theorem. Technical Report CMU-CS-92-152, University of Carnegie Mellon (1992)

10. Myhill, J.: Finite automata and the representation of events. Technical Report WADC TR-57-624, Wright Patterson Air Force Base, Ohio, USA (1957)
11. Nerode, A.: Linear automaton transformations. In: Proceedings of the American Mathematical Society, vol. 9, pp. 541–544. American Mathematical Society (1958)
12. Mezei, J., Wright, J.: Algebraic automata and context-free sets. Information and Control 11, 3–29 (1967)
13. Dezani-Ciancaglini, M., Giovannetti, E., de' Liguoro, U.: Intersection Types, Lambda-models and Böhm Trees. In: MSJ-Memoir. Theories of Types and Proofs, vol. 2, pp. 45–97. Mathematical Society of Japan (1998)
14. Babaev, A.A., Soloviev, S.V.: Coherence theorem for canonical maps in cartesian closed categories. Journal of Soviet Mathematics 20 (1982)
15. de Groote, P., Pogodalla, S.: On the expressive power of abstract categorial grammars: Representing context-free formalisms. Journal of Logic, Language and Information 13(4), 421–438 (2005)
16. Thatcher, J.W.: Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journal of Computer and System Sciences 1(4), 317–322 (1967)
17. Schmidt-Schauß, M.: Decidability of arity-bounded higher-order matching. In: Baader, F. (ed.) CADE 2003. LNCS, vol. 2741, pp. 488–502. Springer, Heidelberg (2003)
18. Comon, H., Jurski, Y.: Higher-order matching and tree automata. In: CSL, pp. 157–176 (1997)
19. Loader, R.: Higher order β matching is undecidable. Logic Journal of the IGPL 11(1), 51–68 (2003)
20. Stirling, C.: A game-theoretic approach to deciding higher-order matching. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 348–359. Springer, Heidelberg (2006)
21. Robertson, N., Seymour, P.D.: Graph minors. v. excluding a planar graph. J. Comb. Theory, Ser. B 41(1), 92–114 (1986)
22. Ong, C.H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS, pp. 81–90. IEEE Computer Society Press, Los Alamitos (2006)

Logic-Based Probabilistic Modeling

Taisuke Sato

Tokyo Institute of Technology, Ookayama Meguro Tokyo Japan
http://sato-www.cs.titech.ac.jp/

Abstract. After briefly mentioning the historical background of PLL/SRL, we examine PRISM, a logic-based modeling language, as an instance of PLL/SRL research. We first look at the distribution semantics, PRISM's semantics, which defines a probability measure on a set of possible Herbrand models. We then mention characteristic features of PRISM as a tool for probabilistic modeling.

1 Introduction

Logic has long been considered as a discipline concerning certainty with major attention on deductive inference. However recent developments in machine learning and other related areas are expanding the role of logic from a vehicle for deductive inference to the one for probabilistic knowledge representation and statistical inference, and has spawned an interdisciplinary subfield of machine learning called PLL (probabilistic logic learning) and/or SRL (statistical relational learning) that combine logic and probability for probabilistic modeling[1].

As its name suggests, PLL[1] has LP (logic programming) and ILP (inductive logic programming) as its backbone. PLL however adds to LP/ILP probability and makes logical formulas random variables taking true and false probabilistically. With such probabilistic formulas we can express logical yet probabilistic phenomena such as gene inheritance cleanly at predicate level. Moreover probabilities statistically learned from real data makes it possible to evaluate how probable a given formula is in the real world.

Furthermore by amalgamating with probability, LP/ILP, or more generally logic, acquires an ability to cope with missing information. Suppose we are going for a picnic tomorrow (P) if the weather is clear (W). Also suppose we have to prepare a lunch box tonight (L) if we go for a picnic tomorrow (P). Then it follows by deduction that we have to prepare a lunch box tonight if the weather is clear tomorrow ($L \Leftarrow W$). However we do not know the truth value of W, tomorrow's weather. Should we prepare a lunch box tonight? In such a case a purely logical inference would completely get stuck as we do not know the truth value of W. The amalgamated system on the other hand will be able to help our decision on whether we should prepare a lunch box or not by looking at the probability associated with W.

[1] SRL seems more often used than PLL now.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 61–71, 2009.

SRL [2], independently of PLL, originated from uncertainty reasoning in BNs (Bayesian networks) [3]. BNs are a graphical representation of joint distributions in terms of directed acyclic graphs. BNs are powerful. First they can visualize complex dependencies among random variables graphically as parents-child nodes. Second they allow us to perform various types of probabilistic inference and parameter learning such as maximum likelihood inference, MAP (maximum a posteriori) inference and Bayes inference. These inferences are efficiently carried out by well-developed algorithms including the celebrated BP (belief propagation) algorithm. Also there are ones for structure learning as well that learn a directed graph from data. As a result BNs are used as one of the standard tools for handling uncertainty and applied to bioinformatics, natural language processing, speech recognition, planning, diagnosis, datamining, decision support and so on.

Nevertheless from a viewpoint of knowledge representation in AI, BNs are primitive. Basic assertions available in BNs are limited to $X = x$ type such that X is a random variable and x is a value. In other words, BNs are a propositional system where $X = x$ is a propositional variable which is true with probability $P(X = x)$. BNs' lack of logical variables, relations and quantifiers, the hallmark of predicate calculus, implies we have no way to express (probabilistic) rules such as "a friend of a friend is a friend with probability 0.3." One might say BNs is a low level language just like machine code.

The low-levelness of BNs prompted the emergence of the KBMC (knowledge-based model construction) approach in the nineties. Initially KBMC built BNs from a knowledge base using a high level language mostly as a matter of convenience. But later it evolved to SRL which upgrades BNs themselves by explicitly introducing variables and relations and/or by embedding them in a richer programming language be it logical or otherwise. Variables in such a language recover an ability to express general patterns. More importantly the introduction of relations opened a new dimension for feature-based probabilistic modeling[2] by using relations themselves as new features [2].

Currently SRL and PLL are moving toward unification but it remains to be seen whether they are eventually unified or not. In this talk, we look into some details of PRISM [4,5,6][3] as an instance of ongoing PLL/SRL research. PRISM is a probabilistic extension of Prolog with a declarative semantics called *distribution semantics* which is a probabilistic generalization of the standard least model semantics of logic programs [7]. Despite the fact that PRISM adopts a single data structure, a single probability computation algorithm and a single parameter learning algorithm, all independent of target models, it universally covers generative models from Bayesian networks to probabilistic grammars with efficiency supported by the PRISM architecture. In what follows we review the distribution semantics in detail and then illustrate some aspects of PRISM from a machine learning point of view.

[2] Features are a function from an input to a real number that characterizes the input. In statistical natural language proceeding for example, often an individual word is used as a feature that returns 1 if the input contains the word, else 0.

[3] `http://sato-www.cs.titech.ac.jp/prism/index.html`

2 Probabilistic Semantics

2.1 The Basic Principle

The distribution semantics, PRISM's declarative semantics, is a probabilistic generalization of the least model semantics in logic programming. Before proceeding however, we look back on a fundamental theorem which underlies it.

When logic and probability are combined, probabilities $P(\varphi)$ are assigned to formulas φ. However since probabilities have to respect Kolmogorov's axioms while formulas obeys their own logical laws, it is not self-evident how to coherently assign probabilities to formulas[4]. We here review Fenstad's representation theorem [8] in our context which states a formal relationship between formulas and their probabilities.

We introduce some notations. Let $\mathcal{L}$ be a countable first order language without equality and ω be a model on a certain fixed domain $\mathbf{U}$[5]. We assume variables are indexed by natural numbers $\mathbf{N}$. We denote by $\varphi[\omega]$ $\{a \in \mathbf{U}^{\mathbf{N}} \mid \omega \models_a \varphi(x_i, \ldots, x_j)\}$ where a is an assignment for the variables such that $a(i)$ is assigned to x_i etc. $\varphi[\omega]$ denotes the set of sequences of elements from $\mathbf{U}$ whose substitution for the variables in φ makes φ true in the model ω.

Theorem 1 (Fenstad 67). *Let φ, ϕ be formulas in $\mathcal{L}$. Suppose probabilities assigned to formulas satisfy the following.*

$$
\begin{aligned}
&(i)\ P(\varphi \vee \phi) + P(\varphi \wedge \phi) = P(\varphi) + P(\phi)\\
&(ii)\ P(\neg\varphi) = 1 - P(\varphi)\\
&(iii)\ P(\varphi) = P(\phi),\ \textit{if}\ \vdash \varphi \leftrightarrow \phi\\
&(iv)\ P(\varphi) = 1,\ \textit{if}\ \vdash \varphi
\end{aligned}
$$

Then there is a σ-additive probability measure λ on the set Ω of models on $\mathbf{U}$ for $\mathcal{L}$. There is also for each $\omega \in \Omega$, a probability μ_ω on the sets of $\varphi[\omega]$'s such that

$$P(\varphi) = \int_S \mu_\omega(\varphi[\omega])\, d\lambda(\omega).$$

When the domain $\mathbf{U}$ is finite and φ is closed, the above theorem reduces to

$$P(\varphi) = \sum_{\omega \models \varphi} \lambda(\{\omega\}).$$

So in this case, the probability is given as the sum of probabilities of the models that satisfy the formula.

Fenstad's theorem strongly suggests to us that if we assign probabilities to formulas in a reasonable way, we should first define a probability measure over models. The *distribution semantics* [4] we explain next follows this idea.

[4] In addition since probabilities are supposed to be used in a probabilistic model, they need to be computable and learnable from data.

[5] $\mathbf{U}$ can be taken as a Herbrand universe for $\mathcal{L}$.

2.2 The Distribution Semantics

Formally a PRISM program DB is a set of definite clauses. We write it as $DB = F \cup R$ where F is a set of ground atoms corresponding to primitive probabilistic events such as coin tossing and R is a set of rules (definite clauses). We assume no atom in F unifies with a head appearing in R.

We consider the set of Herbrand models (we often call them just *worlds* for simplicity) Ω_{DB} for DB and define a probability measure P_{DB} over Ω_{DB} as follows. First enumerate ground atoms in F like $A_1, A_2, \ldots$ and identify a Herbrand interpretation of F with a binary infinite vector $(1, 0, \ldots)$ that specifies A_1 is true (1), A_2 is false (0) and so on. Let $\Omega_F = \prod_i \{0,1\}_i$ be the set of such binary vectors. We give Ω_F a product topology where $\{0,1\}$ has a discrete topology. Also let P_F be a *base* distribution which is any probability measure on the σ-algebra generated by open sets of Ω_F.

Consider an arbitrary subset F' of F. $DB' = F' \cup R$ has the least Herbrand model $\mathbf{M}(DB')$ defined as follows. Let T be the immediate consequence operator [7]. It is applied to a set I of ground atoms and defined by $T(I) = \{A \mid A \Leftarrow B_1 \wedge \cdots \wedge B_h (h \geq 0)$ is a ground instance of a clause in DB' and $\{B_1, \ldots, B_h\} \subseteq I\}$. Put $I_\infty = \cup_{k=0}^{\infty} T^k(\emptyset)$ and verify I_∞ is the least fixpoint of T, i.e. $T(I_\infty) = I_\infty$. Define a Herbrand model $\mathbf{M}(DB')$ by a ground atom A is true in $\mathbf{M}(DB')$ iff $A \in I_\infty$[6]. $\mathbf{M}(DB')$ is called the least Herbrand model of DB'[7]. Using this $\mathbf{M}(DB') = \mathbf{M}(F' \cup R)$ as a model parameterized by F', a subset of F, we can extend P_F by Kolmogorov's extension theorem to a probability measure P_{DB} on the σ-algebra generated by open sets in Ω_{DB} with a product topology (see [5] for details). We consider P_{DB} as the denotation of P_{DB} (distribution semantics).

From the construction of P_{DB}, it is easy to see every closed formula φ is measurable when considered as a function from Ω_{DB} to $\{0,1\}$ such that

$$\varphi(\omega) = \begin{cases} 1 & \text{if } \omega \models \varphi \\ 0 & \text{else} \end{cases}$$

and hence we define the probability $P_{DB}(\varphi)$ of φ as $P_{DB}(\varphi = 1)$. Probabilities thus defined satisfy the conditions from (i) to (iv) of Fenstad's representation theorem. Also we can see the distribution semantics is a generalization of the least model semantics because if the base distribution P_F puts all probability mass on one Herbrand model making $F' \subseteq F$ true, P_{DB} also will put all probability mass on the least Herbrand model of $F' \cup R$.

3 PRISM: From Semantics to Implementation

The *distribution semantics* considers P_{DB} as the denotation of a program $DB = F \cup R$. P_{DB} always exists, uniquely, for any set F of ground atoms, any base measure P_F on F and any set R of definite clauses. Such "semantic robustness"

[6] Proof theoretically $DB' \vdash A$ iff $A \in I_\infty$ for every ground atom A.

[7] In logic programming, the least Herbrand model is considered as the canonical denotation of definite clause programs.

is one of the unique features of PRISM compared to other systems dealing with infinite domains and infinitely many random variables [2,1].

However defining a semantics is one thing and implementing it is another. When implementing the distribution semantics as PRISM as an extension of Prolog, we fix F and restrict the base measure P_F to a denumerable product of of Bernoulli distributions to make P_{DB} computable in probabilistic modeling.

More concretely we introduce ground atoms called `msw` atoms[8] representing a probabilistic choice that take the form $\texttt{msw}(i,v)$ where i is a choice name and v is a chosen value and both are ground terms. We fix the set $F_{\texttt{msw}}$ of ground `msw` atoms and give a (-n infinite) joint distribution $P_{\texttt{msw}}(\cdot)$ in such a way that if a probabilistic choice named i has k choices $v_1,\ldots,v_k$, correspondingly, one of $\texttt{msw}(i,v_1),\ldots,\texttt{msw}(i,v_k)$, say $\texttt{msw}(i,v_j)$ is exclusively true with probability $\theta_{v_j} = P_{\texttt{msw}}(\texttt{msw}(i,v_j))$ ($\sum_j \theta_{v_j} = 1$). $\texttt{msw}(i,v)$ is the only probabilistic built-in predicate in PRISM and used to simulate simple probabilistic events such as coin flipping and dice throwing. The role of definite clauses in R then is to organize such simple events into a complex event corresponding to our observation in the real world. The following is a PRISM program describing the inheritance of ABO blood type. As you see a PRISM program is just liken an ordinary Prolog program[9].

```
values_x(gene,[a,b,o],[0.5,0.2,0.3]).

bloodtype(P) :-
     genotype(X,Y),
     ( X=Y -> P=X ; X=o -> P=Y ; Y=o -> P=X ; P=ab ).
genotype(X,Y) :-
     msw(gene,X), msw(gene,Y).
```

Fig. 1. ABO-blood type program DB_1

The first clause `values_x(gene,[a,b,o],[0.5,0.2,0.3])` is a PRISM declaration specifying $F_{\texttt{msw}}$ and $P_{\texttt{msw}}(\cdot)$. It introduces a set of mutually exclusive atoms $\{\texttt{msw}(\texttt{gene},\texttt{a}), \texttt{msw}(\texttt{gene},\texttt{b}), \texttt{msw}(\texttt{gene},\texttt{o})\}$ corresponding to a probabilistic choice named `gene` having three possible outcomes `a`, `b` and `o`, representing three genes determing one's ABO blood type. They are true with 0.5, 0.2 and 0.3 respectively as indicated by the `values_x` declaration. The second clause is a rule specifying the relationship between genotypes (pair of genes) and phenotypes (ABO blood type, `a`, `b`, `o`, `ab`). For example if a genotype is $(\texttt{a},\texttt{b})$, the blood type is `ab`. The last clause simulates the inheritance of two genes, one from each parent. Sampling `msw(gene,X)` returns $\texttt{X} = \texttt{a}$ with probability 0.5 etc. In PRISM, textually different occurrences of `msw` atoms in a program are treated

[8] `msw` stands for "multi-ary random switch."
[9] Some familiarity with Prolog is assumed here.

as independent. So `msw(gene,X)` and `msw(gene,Y)` are independent. This program as a whole describes how `bloodtype(P)`, our observation, is generated by a sequential choices made by `msw(gene,·)` atoms.

Once loaded into computer memory by the PRISM system, DB_1 can answer various questions such as the probability of `bloodtype(a)`. PRISM computes it by way of search and the resulting propositional AND/OR formula called an explanation graph. To be precise it first performs an exhaustive SLD search for `?- bloodtype(a)` and collects all conjunctions E_1, E_2 and E_3 such that $E_i, DB_1 \vdash \texttt{bloodtype(a)}$ $(i = 1, 2, 3)$ where $E_1 = \texttt{msw(gene, a)} \wedge \texttt{msw(gene, a)}$, $E_2 = \texttt{msw(gene, a)} \wedge \texttt{msw(gene, o)}$ and $E_3 = \texttt{msw(gene, o)} \wedge \texttt{msw(gene, a)}$. We call each E_i an *explanation* for `bloodtype(a)`. Since the search is exhaustive, $\texttt{bloodtype(a)} \Leftrightarrow E_1 \vee E_2 \vee E_3$ holds with probability one in terms of PRISM's semantics. In addition since E_1, E_2 and E_3 are obtained by mutually exclusive proof paths, they are mutually exclusive as well. Also recall that `msw` atoms are independent. Putting these together $P_{DB_1}(\texttt{bloodtype(a)})$ is calculated as

$$\begin{aligned} P_{DB_1}(\texttt{bloodtype(a)} \mid \theta_a, \theta_b, \theta_o) &= P_{DB_1}(E_1 \vee E_2 \vee E_3) \\ &= P_{DB_1}(E_1) + P_{DB_1}(E_2) + P_{DB_1}(E_3) \\ &= \theta_a^2 + \theta_a\theta_o + \theta_o\theta_a \\ &= 0.45 \end{aligned}$$

where $\theta_a = P_{\texttt{msw}}(\texttt{msw(gene, a)}) = 0.5$, $\theta_b = P_{\texttt{msw}}(\texttt{msw(gene, b)}) = 0.2$ and $\theta_o = P_{\texttt{msw}}(\texttt{msw(gene, o)}) = 0.3$[10] as specified by `values_x(gene,[a,b,o],[0.5, 0.2,0.3])`.

4 Statistical Abduction

Although PRISM is an extension of Prolog, their inferences are of different type. Prolog is a logical language for (controlled) deduction whereas PRISM is a logical language for (controlled) abduction. In general abduction refers to "inference to the best explanation." Given a knowledge base K, a set of formulas, and an observation O, we seek for the best explanation E in abduction such that $K \wedge E \vdash O$ and $K \wedge E$ is consistent. The exhaustive search for explanations for the given goal in PRISM is exactly an abductive inference. One of the problems in abduction is that there can be many explanations just like a student has many excuses for not doing homework. In the blood type example, PRISM abduces three explanations E_1, E_2, and E_3 for the observation `bloodtype(a)` but in the case of parsing where observations are sentences, the knowledge base is a grammar and an explanations is a parse tree, we often have tens of thousands of parse trees for one sentence. In the face of multiple explanations, we need to choose somehow one of them as the best one.

[10] $P_{DB_1}(\cdot)$ is an extension of $P_{\texttt{msw}}(\cdot)$, so $P_{DB_1}(E_1) = P_{\texttt{msw}}(\texttt{msw(gene, a)} \wedge \texttt{msw(gene, a)}) = P_{\texttt{msw}}(\texttt{msw(gene}, a))^2 = {\theta_a}^2$.

Statistical abduction [9] resolves the problem of multiple explanations that arises in abduction by introducing a probabilistic model $P(\cdot)$ connecting explanations E, a knowledge base K and an observation O. Using $P(\cdot)$ we choose the most probable E giving the highest $P(E \mid O, K)$ such that $K \wedge E \vdash O$. In this sense PRISM is not just a language for abduction but a language for statistical abduction, and indeed the first one with an ability to perform statistical inference to our knowledge. In PRISM the knowledge base is a program DB for which the distribution semantics guarantees $P_{DB}(\mathrm{iff}(DB)) = 1$[11] [5]. Accordingly

$$\begin{aligned} E^* &= \mathrm{argmax}_E P_{DB}(E \mid O, \mathrm{iff}(DB)) \\ &= \mathrm{argmax}_E P_{DB}(E \wedge O \wedge \mathrm{iff}(DB)) \\ &= \mathrm{argmax}_E P_{DB}(E \wedge \mathrm{iff}(DB)) \\ &= \mathrm{argmax}_E P_{DB}(E) \end{aligned}$$

holds. Thus seeking for the best explanation in statistical abduction is equivalent to Viterbi inference implemented in PRISM, giving $E^* = E_1$ as the best explanation for `bloodtype(a)`.

5 Probabilistic Modeling

So far we have been looking at theoretical aspects of PRISM. Here we examine PRISM as a practical tool for probabilistic modeling. As a modeling tool, the most salient feature of PRISM is model specification by (recursive) definite clauses[12], which results in

- universality (for generative models and their parameter learning)
- high level specification (small amount of coding) and
- interpretability (what the system does is readable to humans).

The first point is due to the fact that PRISM can simulate, as an extension of Prolog, a non-deterministic Turing machine in which non-determinacy is resolved by a probabilistic choice, and in addition, PRISM has a generic routine (the graphical EM algorithm [5]) for parameter learning. The universality covers PCFGs (probabilistic context free grammars) [11] as well as BNs [3]. A PCFG is a CFG with probabilities assigned to grammar rules. It generates a sentence by repeatedly making a probabilistic choice of a grammar rule and expanding a nonterminal with it until no nonterminal remains. Since there is no upper limit on the number of applications of grammar rules, if there is a recursive rule, we use a countably many iid random variables. Hence finite probabilistic models

[11] $\mathrm{iff}(DB)$ is the if-and-only-if completion of DB. Definite clauses with a common head such as $A \Leftarrow B$ and $A \Leftarrow C$ in a program are lumped together to the if-and-only-if form $A \Leftrightarrow B \vee C$ in $\mathrm{iff}(DB)$.

[12] Actually general clauses (those that may contain negative goals in the clause body) are allowed under a certain condition [10].

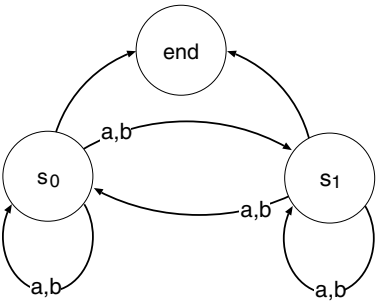


Fig. 2. An HMM

```
values_x(init,[s0,s1],[0.5,0.5]).
values_x(out(_),[a,b],[0.5,0.5]).
values_x(trans(_),[s0,s1,end],[0.7,0.2,0.1]).

hmm(L):- msw(init,S0),hmm(S0,L).
hmm(S,L):- msw(trans(S),NextS),
     (NextS=end -> L=[] ; msw(out(S),C), L=[C|Cs], hmm(NextS,Cs)).
```

Fig. 3. $DB_{\texttt{hmm}}$ for the HMM in Fig.2

such as BNs or otherwise cannot express PCFGs though they are the most basic class of probabilistic grammars [12].

The second point owes to the power of first-order expressions such as variables, terms, relations and recursion. For example HMMs (hidden Markov models) which are a class of stochastic automata very popular in machine learning can be expressed in three lines (together with three line declarations) as shown in Fig. 3.

We hope the program $DB_{\texttt{hmm}}$ in Fig. 2 is self-explanatory but add comments[13]. This program generates indefinitely long lists of `a` and `b`. There are two states `s0` and `s1` both of which can be an initial state. After choosing the initial state, it goes into infinite recursion, while outputting `a` or `b` on transition, until a choice of transition to `end`, the final state, is made probabilistically.

Fig. 4 is a sample session of $DB_{\texttt{hmm}}$. After loading, we issue `?-prob(hmm([a,a,b]))` to compute the probability of `hmm([a,a,b])`. Next we ask what is the most probable state transition sequence (Viterbi inference) for generating `hmm([a,a,b])`. The answer is given as a calling sequence (in Prolog) of subgoals. Finally, we learn parameters from a list of observations {`hmm([a,a,b])`, `hmm([b,a])`, `hmm([b,b])`, `hmm([a,b,a])`} by invoking the graphical EM algorithm using `learn/1`. After 18 iterations it converged with log-likelihood −12.908717468, giving parameters as listed. "`Switch init`" signifies the parameters are for `msw(init,·)`.

[13] `values_x(out(_),[a,b],[0.5,0.5])` is a template where the underscore `_` can be replaced with any term.

```
?- prob(hmm([a,a,b])).
Probability of hmm([a,a,b]) is: 0.012345679012346

?- viterbif(hmm([a,a,b])).
hmm([a,a,b]) <= hmm(s0,[a,a,b]) & msw(init,s0)
hmm(s0,[a,a,b]) <= hmm(s1,[a,b]) & msw(trans(s0),s1) & msw(out(s0),a)
hmm(s1,[a,b]) <= hmm(s1,[b]) & msw(trans(s1),s1) & msw(out(s1),a)
hmm(s1,[b]) <= hmm(s1,[]) & msw(trans(s1),s1) & msw(out(s1),b)
hmm(s1,[]) <= msw(trans(s1),end)

?- learn([hmm([a,a,b]),hmm([b,a]),hmm([b,b]),hmm([a,b,a])]).
...
#em-iterations: 0.(18) (Converged: -12.908717468)
Switch init: s0 (p: 0.000005604) s1 (p: 0.999994396)
Switch out(s0): a (p: 0.499954371) b (p: 0.500045629)
Switch out(s1): a (p: 0.500019760) b (p: 0.499980240)
Switch trans(s0): s0 (p: 0.430352826) s1 (p: 0.000001065) end (p: 0.569646108)
Switch trans(s1): s0 (p: 0.573218835) s1 (p: 0.426780543) end (p: 0.000000623)
```

Fig. 4. Running $DB_{\texttt{hmm}}$

We remark that it is straightforward to extend and modify, say merge with a PCFG, the above skeletal program to one's purpose. When the user modifies his probabilistic model, it often happens that he has to start from designing a new data structure all over again. Since PRISM adopts a single data structure (explanation graphs), there is no need for a new data structure. All you need to change when you change your model is the specification part alone, which is a labor saving aspect of PRISM.

The third point, interpretability, is of particular importance in practice. Eventually the outcome of our analysis by probabilistic modeling must be transferred to non-experts. However think of non-generative probabilistic models specified by "weights" w_i like $P(y \mid x) \propto \exp(\sum_i w_i f_i(x, y))$. It would be very hard to explain the meaning of those weights to non-experts, and especially so when there are a huge number of weights like in statistical natural language processing. On the contrary, logic-based probabilistic modeling uses logical formulas to specify models, which seem more readable and more meaningful to non-experts than weights, though we admit logical formulas themselves might be an obstacle.

Last but not least, we comment on complexity. PRISM is a high-level modeling language and models can be described succinctly as a PRISM program. However one might ask if the ease of modeling sacrifices efficiency. The answer is possibly so but marginally. Due to the complexity analysis of representative models [5], HMMs, PCFGs and BNs can be computed in the same time complexity as specialized algorithms (the Baum-Welch algorithm for HMMs, the Inside-Outside algorithm for PCFGs, BP for BNs [13]). The reason is that explanation graphs, PRISM's data structure, realize structure-sharing and probabilities and

expectations (needed for parameter learning) are computed by dynamic programming exploiting such structure-sharing. The real issue here is the trade off between general data structure for every model and specialized data structure for a specific model. PRISM lies on the general end of this scale. It constructs explanation graphs using pointers as their size depends on a model and data and is unknown beforehand in general. This causes a disadvantage in computational efficiency. We believe however the flexibility can compensate for such a disadvantage and implementation efforts can make it minimal.

6 Concluding Remarks

We reviewed the historical background of PLL/SRL and examined PRISM as an instance of PLL/SRL research. It is a logic-based modeling language we have been developing in the past decade and provides a general tool for generative modeling in machine learning. As an extension of Prolog, it subsumes Prolog and furthermore has the ability to learn parameters from data based on an abductive framework called "statistical abduction." We omitted most of computational details but they can be reached by [4,5,14,6]. Also omitted is variational Bayes which is the latest feature of PRISM for Bayesian inference [15].

References

1. De Raedt, L., Kersting, K.: Probabilistic inductive logic programming. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.) Probabilistic Inductive Logic Programming. LNCS, vol. 4911, pp. 1–27. Springer, Heidelberg (2008)
2. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press, Cambridge (2007)
3. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco (1988)
4. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP 1995), pp. 715–729 (1995)
5. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. Journal of Artificial Intelligence Research 15, 391–454 (2001)
6. Sato, T., Kameya, Y.: New Advances in Logid-Based Probabilistic Modeling by PRISM. In: De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.) Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911, pp. 118–155. Springer, Heidelberg (2008)
7. Lloyd, J.W.: Foundations of Logic Programming. Springer, Heidelberg (1984)
8. Fenstad, J.E.: Representation of probabilities defined on first order languages. In: Crossley, J.N. (ed.) Sets, Models and Recursion Theory, pp. 156–172. North-Holland, Amsterdam (1967)
9. Sato, T., Kameya, Y.: Statistical abduction with tabulation. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 567–587. Springer, Heidelberg (2002)
10. Sato, T., Kameya, Y., Zhou, N.F.: Generative modeling with failure in PRISM. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), pp. 847–852 (2005)

11. Wetherell, C.S.: Probabilistic languages: a review and some open questions. Computing Surveys 12(4), 361–379 (1980)
12. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
13. Sato, T.: Inside-Outside probability computation for belief propagation. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 2605–2610 (2007)
14. Zhou, N.F., Sato, T., Shen, Y.D.: Linear tabling strategies and optimization. Theory and Practice of Logic Programming 8(1), 81–109 (2008)
15. Sato, T., Kameya, Y., Kurihara, K.: Variational bayes via propositionalized probability computation in prism. Annals of Mathematics and Artificial Intelligence (to appear)

Completions of Basic Algebras

Majid Alizadeh*

School of Mathematics, Statistics and Computer Science, College of Science, University of Tehran, P. O. Box 14155-6455, Tehran, Iran
Research center for Integrated Science, Japan Advanced Institute of Science and Technology, Asahidai, Nomi, Ishikawa, 923-1292, Japan

Abstract. We discuss completions of basic algebras. We prove that the ideal completion of a basic algebra is also a basic algebra. It will be shown that basic algebras are not closed under MacNeille completions. By adding the join-infinite distributive law to basic algebras, we will show that these kind of basic algebras are closed under the closed ideal completion and moreover any other regular completions of these algebras are isomorphic to the closed ideal completion. As an application we establish an algebraic completeness theorem for a logic weaker than Visser's basic predicate logic, BQL, a proper subsystem of intuitionistic predicate logic, IQL.

Keywords: Heyting algebra, Basic algebra, Completion, Visser's basic logic, Intuitionistic logic.

1 Introduction

In the present paper, we will discuss completions of algebras for Visser's basic logic and a completeness theorem for it which is obtained using completions. Completions of basic algebras and some of its subvarieties are discussed in the second and third sections. In the second section, we will first give a brief overview of the canonical extensions of basic algebras and then we will introduce the ideal completion of a basic algebra. We will see that the ideal completion is also a Heyting algebra and consequently we can embed every basic algebra into a complete Heyting algebra as a lattice. None of these completions of basic algebras are regular, i.e., the embedding of a basic algebra into these two kinds of complete algebras does not preserve existing infinite joins and meets in general. In the third section, we first show that the variety of basic algebras, a subvariety of this variety called Löb algebras and all the subvarieties of the latter are not closed under MacNeille completions. Then, we introduce the closed ideal completion of a basic algebra satisfying the join-infinite distributive law and show that the

* I would like to thank Prof. Mohammad Ardeshir for his valuable comments. This paper was completed during my stay at Japan Advanced Institute of Science and Technology, JAIST, as a visiting researcher. I am grateful to Prof. Hiroakira Ono for his invitation to my attendance at JAIST also for reading early draft of the paper and giving helpful comments and suggestions.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 72–83, 2009.

class of basic algebras which satisfies the join-infinite distributive law is closed under this completion and moreover any other regular completions of these kinds of algebras are isomorphic to the closed ideal completion. In the last section, we apply the regular completion of basic algebras that satisfy the join-infinite distributive law to prove an algebraic completeness theorem for a logic weaker than Visser's basic predicate logic.

2 Canonical Completion and Ideal Completion

In this section first we give a brief overview of canonical extension of basic algebras and then we will introduce the ideal completion of these algebras.

Definition 1. *A* basic algebra $\mathbf{B} = \langle B, \wedge, \vee, \rightarrow, 0, 1\rangle$ *is a structure with constants* 0 *and* 1*, and binary functions* $\wedge$*,* $\vee$*, and* $\rightarrow$*, such that*

1. *with respect to* 0, 1, $\wedge$*, and* $\vee$ *we have a bounded distributive lattice, and*
2. *for* $\rightarrow$ *we have the additional identities and quasi-identities*

$$a \rightarrow b \wedge c = (a \rightarrow b) \wedge (a \rightarrow c);$$

$$b \vee c \rightarrow a = (b \rightarrow a) \wedge (c \rightarrow a);$$

$$a \rightarrow a = 1;$$

$$a \leq 1 \rightarrow a; \quad \textit{and}$$

$$(a \rightarrow b) \wedge (b \rightarrow c) \leq a \rightarrow c.$$

The relation $\leq$ is expressible in term of equations with $\wedge$ or $\vee$ in the standard way, i.e.,

$$a \leq b \quad \text{iff} \quad a \wedge b = a \quad \text{iff} \quad a \vee b = b.$$

So this class of algebras forms a variety. A basic algebra $\mathbf{B}$ is called a *Löb algebra* iff for all $x \in B$, $(1 \rightarrow x) \rightarrow x = 1 \rightarrow x$.

Lemma 1. *[1]. Let* $\mathbf{B}$ *be a basic algebra. Then for* $a, b, \in B$*,*

1. *if* $a \leq b$*, then* $a \wedge (b \rightarrow c) = a \wedge (1 \rightarrow c)$*,*
2. *if* $a \leq b$*, then* $b \rightarrow c \leq a \rightarrow c$*, and* $c \rightarrow a \leq c \rightarrow b$*,*
3. *if* $a \wedge b \leq c$*, then* $a \leq b \rightarrow c$*.*

A basic algebra $\mathbf{B}$ is called *complete* if it is complete as a lattice. A variety of basic algebras is closed under a completion method if the completion of every algebra in the variety according to the method belongs to the variety too.

Canonical extensions of basic algebras, introduced by Ardeshir [4], are briefly described as follows. For a basic algebra $\mathbf{B}$, let $W = W_{\mathbf{B}}$ be the set of all prime filters of $\mathbf{B}$. Define a binary relation $\prec$ on W as follows: $F \prec F^{'}$ iff $b \in F^{'}$ whenever $a \rightarrow b \in F$ and $a \in F^{'}$. A subset X of W is called an *upset* of W if and only if for each $F, G \in W$ such that $F \in X$ and $F \subseteq G$, we have $G \in X$. Then the structure $\langle UP(W), \cap, \cup, \rightarrow, \emptyset, W\rangle$ of upsets, $UP(W)$, of W is a basic algebra, where

$$X \to Y = \{F \in W \ : \ \forall G \succ F(\text{if } \ G \in X \ \text{ then } \ G \in Y)\}.$$

One can embed the original algebra $\mathbf{B}$ in its canonical extension by a mapping f defined by

$$f(a) = \{P \mid P \text{ is a prime filter and } a \in P\}, \ \text{ for each } a \in B.$$

It is well-known that the above canonical embedding works well for distributive lattices but it does not preserve existing infinite joins and meets in general. In the rest of this section we introduce the ideal completion of basic algebras. First, let us recall some known definitions.

A subset $I \subset B$ is an *ideal* on $\mathbf{B}$ if $0 \in I$; if $a, b \in I$, then $a \vee b \in I$ and if $a \leq b$ and $b \in I$, then $a \in I$. An ideal I on $\mathbf{B}$ is called *prime*, if $a \wedge b \in I$ implies that $a \in I$ or $b \in I$. As usual we use the symbol $(a]$ for the ideal generated by the element a of the basic algebra $\mathbf{B}$, i.e., $(a] = \{x \in B : \ \ x \leq a\}$. For $S \subseteq B$, put $I(S) = (S] = \{x \in B : \ \ x \leq (a_1 \vee ... \vee a_n)$ for some elements $a_1, ..., a_n \ \ in \ S\}$. It is easy to show that $(S]$ is the minimal ideal containing S. For ideals I and J in $\mathbf{B}$ we define $I \wedge J = I \cap J$. For a set Γ of ideals of $\mathbf{B}$, $\vee\Gamma$ is defined by $(\cup\Gamma]$, in particular $I \vee J = (I \cup J]$.

For a basic algebra $\mathbf{B}$ the set $I(\mathbf{B})$ of all ideals of $\mathbf{B}$ forms a complete distributive lattice with respect to $\wedge$ and $\vee$. We call it *the ideal completion* of $\mathbf{B}$.

Theorem 1. *Let* $\mathbf{B}$ *be a basic algebra. Then* $I(\mathbf{B}) = \langle I(B), \wedge, \vee, \to^*, (0], (1]\rangle$ *where, for* $I, J \in I(B)$,

$$I \to^* J = \{x \in B : \textit{for every } i \textit{ in } I \textit{ there is a } j \textit{ in } J \textit{ such that } x \leq i \to j\}$$

is a basic algebra.

Proof. First, we show that $I \to^* J$ is an ideal, for every ideal I and J. Easily one can see that $0 \in I \to^* J$ also if $x \leq y$ and $y \in I \to^* J$, then $x \in I \to^* J$. Now let $x, y \in I \to^* J$, we show that $x \vee y \in I \to^* J$. Let $i \in I$, then there are $j_1, j_2 \in J$ such that $x \leq i \to j_1$ and $y \leq i \to j_2$. So, $x \vee y \leq (i \to j_1) \vee (i \to j_2) \leq i \to j_1 \vee j_2$. Thus $x \vee y \in I \to^* J$.

It is clear that for every ideal I and J, $J \subseteq I \to^* J$. For ideals I, J and K we only show that, $I \vee J \to^* K = (I \to^* K) \wedge (J \to^* K)$.

Let $x \in (I \to^* K) \wedge (J \to^* K)$ and $u \in I \vee J$, then there exist $a_1, ..., a_t \in I$ and $b_1, ..., b_s \in J$ such that, $u \leq (\bigvee_{i=1}^{t} a_i) \vee (\bigvee_{i=1}^{s} b_i)$. Let $a = \bigvee_{i=1}^{t} a_i$ and $b = \bigvee_{i=1}^{s} b_i$, then $a \in I$ and $b \in J$, so there exist $v_1, v_2 \in K$ such that, $x \leq a \to v_1 \leq a \to v_1 \vee v_2$ and $x \leq b \to v_2 \leq b \to v_1 \vee v_2$. Then $x \leq a \vee b \to v_1 \vee v_2 \leq u \to v_1 \vee v_2$. Hence $x \in I \vee J \to^* K$. The converse is trivial and by semilar arguments we can prove the others.

Theorem 2. *Let* $\mathbf{B}$ *be a basic algebra. Then the map* $x \longmapsto (x]$ *embeds* $\mathbf{B}$ *into the complete basic algebra* $I(\mathbf{B})$.

Proof. Clearly $x \longmapsto (x]$ is one to one and preserves $\wedge, \vee, 0$ and 1. We show that it also preserves " $\to$ ". Let $a \in (x] \to^* (y]$, then for element $u \leq x$, there is an element $v \leq y$ such that $a \leq u \to v$. Suppose that $u = x$, then

$a \leq x \to v \leq x \to y$, therefore $a \in (x \to y]$ and so $(x] \to^* (y] \subseteq (x \to y]$. The converse is obvious.

In the following theorem we show the relation between the ideal completion and Heyting algebras

Proposition 1. *The basic algebra $I(\mathbf{B})$ is a Heyting algebra if and only if $\mathbf{B}$ is a Heyting algebra.*

Proof. It is known that a basic algebra $\mathbf{B}$ is a Heyting algebra if and only if for any element x we have $1 \to x = x$, see [1]. On the other hand, by Theorem 1, for any element x in basic algebra $\mathbf{B}$ we have $(1 \to x] = (1] \to^* (x]$. Now we can get the desired result.

In the following theorem we give some basic properties of the ideal completion of basic algebras.

Theorem 3. *Let $\mathbf{B}$ be a basic algebra. Then $I(\mathbf{B})$ satisfies the following properties:*

1. $I \wedge \bigvee_{t\in T} J_t = \bigvee_{t\in T}(I \wedge J_t)$,
2. $I \vee \bigwedge_{t\in T} J_t \leq \bigwedge_{t\in T}(I \vee J_t)$,
3. $\bigvee_{t\in T}(I_t \to^* J) \leq \bigwedge_{t\in T} I_t \to^* J$,
4. $\bigvee_{t\in T}(I \to^* J_t) \leq I \to^* \bigvee_{t\in T} J_t$,
5. $\bigwedge_{t\in T}(I_t \to^* J) = \bigvee_{t\in T} I_t \to^* J$,
6. $\bigwedge_{t\in T}(I_t \to^* J_t) \leq \bigvee_{t\in T} I_t \to^* \bigvee_{t\in T} J_t$.

Proof. We only prove clauses 1 and 5. Similar arguments work for the others.

1. Let $x \in I \wedge \bigvee_{t\in T} J_t$, then there exist elements $a_1, \ldots, a_n \in \bigcup_{t\in T} J_t$ such that $x \leq \bigvee_{i=1}^n a_i$. Without loss of generality we can assume that $a_i \in J_i$, so $x \in \bigvee_{i=1}^n J_i$. By distributivity of $I(\mathbf{B})$, we have $x \in \bigvee_{i=1}^n (I \wedge J_i)$, so $x \in \bigvee_{t\in T}(I \wedge J_t)$. The converse is trivial.

5. Let $x \in \bigwedge_{t\in T}(I_t \to^* J)$, then for every $t \in T$, $x \in I_t \to^* J$. Let $u \in \bigvee_{t\in T} I_t$, then there exist elements $a_1, ..., a_n$, such that $u \leq \bigvee_{i=1}^n a_i$. Without loss of generality we can assume that $a_i \in I_i$. For each a_i there is b_i in J such that $x \leq a_i \to b_i \leq a_i \to \bigvee_{i=1}^n b_i$, then $x \leq (a_1 \to \bigvee_{i=1}^n b_i) \wedge ... \wedge (a_n \to \bigvee_{i=1}^n b_i) \leq u \to \bigvee_{i=1}^n b_i$. So $x \in \bigvee_{t\in T} I_t \to^* J$, since $\bigvee_{i=1}^n b_i \in J$. On the other hand, we have $I_t \leq \bigvee_{t\in T} I_t$, so $\bigvee_{t\in T} I_t \to^* J \leq I_t \to^* J$ and hence $\bigvee_{t\in T} I_t \to^* J \leq \bigwedge_{t\in T}(I_t \to^* J)$.

Definition 2. *A lattice $\mathbf{L}$ is called join-infinite distributive, if the equality $a \wedge (\bigvee_t b_t) = \bigvee_t (a \wedge b_t)$ holds in the case that $\bigvee_t b_t$ exists.*

It is well-known that a complete bounded distributive lattice is a Heyting algebra if and only if it is join-infinite distributive. From Theorem 2 and 3 it follows that the ideal completion of every basic algebra $\mathbf{A}$ has (implicity) a structure of a Heyting algebra. In this algebra the Heyting implication is the operation defined for every a, b by:

$$a \hookrightarrow b = \bigvee \{c : a \wedge c \leq b\}$$

This implication may be different from the basic algebra implication defined in Theorem 1, as the following example shows.

Example 1. Let $\mathbf{B}$ be a bounded distributive lattice with universe $B = \{0, a, b, 1\}$ such that $a \wedge b = 0$, $a \vee b = 1$. We define an implication on B as follows: $1 \to a = b \to a = a$, $1 \to b = a \to b = 1$, $x \to 0 = a$, for $x > 0$ and $x \to y = 1$, for $x \leq y$. It is easy to see that $\langle \mathbf{B}, \to \rangle$ is a basic algebra.

Now the ideal completion of $\mathbf{B}$ is a finite distributive lattice, so it is a Heyting algebra too. In this case we have $B \hookrightarrow^* (b] = (b]$. But by basic algebra implication defined in Theorem 1 we have $B \to^* (b] = B$.

Theorem 2 implies only the following:

Theorem 4. *Every basic algebra can be embedded in a complete Heyting algebra as a lattice.*

Similar to canonical extensions of basic algebras, also the ideal completion of these algebras does not preserve the existing infinite joins and infinite meet in general. In fact, it does not preserve infinite joins. We end this section by a theorem which says that the variety of Löb algebras is not closed under the ideal completion. First we note the following

Lemma 2. *[2]. Let* $\mathbf{B}$ *be a Löb algebra. Then for any element* $x \in B$, $1 \to x = x$ *if and only if* $x = 1$.

Theorem 5. *The variety of Löb algebras is not closed under the ideal completion.*

Proof. Take the linear order structure $\langle N + N^*, < \rangle$, where N is the set of all natural numbers with the strict order, followed by a copy of N with reverse order, i.e., $\{0 < 1 < 2 < \cdots < 2^* < 1^* < 0^*\}$. Let $\mathbf{B}$ be the algebra with universe $N + N^*$ and with greatest and least elements 0^* and 0, respectively. For each element $x \in N - \{0\}$, the successor $s(x^*)$ of x^* is defined to be $(x - 1)^*$. For every element x, y in B, $x \to y$ is equal to $s(y)$, if $y < x$ and 0^* otherwise.

It can easily be seen that this structure is a Löb algebra. Now for ideal $N = (N]$ in the ideal completion of $\mathbf{B}$ we have $B \to^* N = N$. So by pervious Lemma the ideal completion of $\mathbf{B}$ is not a Löb algebra.

3 MacNeille Completions and Closed Ideal Completions

In the previous section we saw that the canonical extension and the ideal completion of a basic algebra do not necessarily preserve existing infinite joins and meets in the original algebra. In this section we will discuss two other completions of basic algebras, the MacNeille completion and the closed ideal completion. It is well-known that the MacNeille completion of a distributive lattice preserves existing infinite joins and meets but does not necessarily preserve distributivity. But MacNeille completions of Heyting algebras are still Heyting algebras.The MacNeille completion of a distributive lattice is briefly described as follows. For a distributive lattice $\mathbf{A}$ and $B \subseteq A$, let $L(B)$ be the collection of all lower bounds of B, $U(B)$ be the collection of all upper bounds of B, and call B a normal ideal

of $\mathbf{A}$ if $B = LU(B)$. Then the collection of all normal ideals of $\mathbf{A}$ is the MacNeille completion of $\mathbf{A}$. Unlike the variety of Heyting algebras, in the following theorem we will see that the variety of basic algebras is not closed under MacNeille completions.

Theorem 6. *The variety of basic algebras is not closed under MacNeille completions.*

Proof. It is well-known [6] that the variety of bounded distributive lattices is not closed under MacNeille completions. Now suppose that $\mathbf{L}$ is a bounded lattice which is distributive and $\rightarrow$ is defined as the function constantly equal to 1. Then $\langle \mathbf{L}, \rightarrow \rangle$ is a basic algebra, indeed a Löb algebra. Now if $\mathbf{L}$ is a distributive lattice whose MacNeille completion is not distributive, then independently of how one extends implication, it is clear that the result can not be a basic algebra.

A basic algebra $\mathbf{B}$ is called an $\mathcal{L}^1$-algebra if it satisfies $1 \rightarrow 0 = 1$. Note that in every basic algebra $\mathbf{B}$, $1 \rightarrow 0 = 1$ if and only if for any element a and b in $\mathbf{B}$ we have $a \rightarrow b = 1$. It is easy to see that every $\mathcal{L}^1$-algebra is a Löb algebra. In [3] it was shown that the minimal subvarieties of the variety of basic algebras are only the variety of Boolean algebras and the variety of $\mathcal{L}^1$-algebras, and consequently the latter is the single minimal variety of the variety of Löb algebras. Two famous varieties of basic algebras are the variety of Heyting algebras and the variety of Löb algebras and all of their subvarieties. It was shown [7] that the only varieties of Heyting algebras which are closed under MacNeille completions are the trivial variety, the variety of Boolean algebras, and the variety of Heyting algebras. We have the following theorem for the variety of Löb algebras.

Theorem 7. *1. If a subvariety of basic algebras contains the two elements $\mathcal{L}^1$-algebra, then it is not closed under MacNeille completions.*
2. The variety of Löb algebras and all of its subvarieties are not closed under MacNeille completions.

Proof. It was shown [3] that the two elements $\mathcal{L}^1$-algebra generates the variety of all $\mathcal{L}^1$-algebras. Now by using the same argument of Theorem 6 one can get the desired result. 2 is a corollary of 1.

Let us recall some definitions to introduce the *closed ideal completion* for basic algebras. An ideal I of a basic algebra $\mathbf{B}$ is *closed* if for every subset S in I we have $\bigvee S \in I$, whenever $\bigvee S$ exists. Suppose that $I_c(\mathbf{B})$ is the set of closed ideals of $\mathbf{B}$. For $\Gamma \subseteq I_c(\mathbf{B})$ we define $\bigvee^c \Gamma = I_c(\bigcup \Gamma)$, i.e., the minimal closed ideal including $\bigcup \Gamma$. Note that for every subset X of a basic algebra, there exists a unique minimal closed ideal $I_c(X)$ including X. In fact $I_c(X)$ is the intersection of all closed ideals including X. For $I, J \in I_c(\mathbf{B})$ we define $I \rightarrow^c J = I_c(I \rightarrow^* J)$. Note that in any Heyting algebra, according to the presence of the law of residuation, i.e.,

$$a \wedge b \leq c \quad \text{if and only if} \quad a \leq b \rightarrow c$$

we can show that an ideal is normal if and only if it is closed under existing joins. So in the variety of Heyting algebras, MacNeille completions and closed ideal completions coincide. We will show that the analogous result is true for some kind of basic algebras. For convenience we include a proof of the following known result.

Proposition 2. *Let* $\mathbf{B}$ *be a join-infinite distributive basic algebra and* $X \subseteq B$. *Then every element* x *of* $I_c(X)$ *is characterized as follows:*

$$x = \bigvee\{y \mid y \leq x \text{ and } y \leq z \text{ for some } z \in X\}.$$

Proof. Let J be the set of all x's in the proposition. First we show that it is an ideal. Suppose that $x \in J$ and $a \leq x$, then by the following sequences of identities we can get $a \in J$.

$$\begin{aligned} a = a \wedge x &= a \wedge \bigvee\{y \mid y \leq x \text{ and } y \leq z \text{ for some } z \in X\} \\ &= \bigvee\{y \wedge a \mid y \leq x \text{ and } y \leq z \text{ for some } z \in X\} \\ &= \bigvee\{y \mid y \leq a \text{ and } y \leq z \text{ for some } z \in X\}. \end{aligned}$$

In the third identity we used the join-infinite distributive law. It is fairly easy to show that if x and y are in J, then $x \vee y$ is also in J. Now assume that S is a subset of J and $a = \bigvee\{s \mid s \leq a \text{ and } s \in S\}$. For $s \in S$ put $A_s = \{x \mid x \leq s \text{ and } x \leq y \text{ for some } y \in X\}$. Then $s = \bigvee A_s$ for s in S. So

$a = \bigvee\{\bigvee A_s \mid s \leq a \text{ and } s \in S\} = \bigvee\{x \mid x \leq a \text{ and } x \leq y \text{ for some } y \in X\}$. Therefore a is in J. J also includes X and the minimality of J is clear.

Let us recall that an embedding of an algebra $\mathbf{B}$ to a complete algebra $\mathbf{B}^*$ is called *regular embedding* if it preserves the existing infinite joins and meets in $\mathbf{B}$. In this case the complete algebra $\mathbf{B}^*$ is called a *regular completion* of $\mathbf{B}$.

Theorem 8. *Let* $\mathbf{B}$ *be a join-infinite distributive basic algebra. Then*

$$I_c(\mathbf{B}) = \langle I_c(B), \wedge, \vee^c, \to^c, (0], (1]\rangle$$

is a join-infinite and complete basic algebra and the embedding $f : \mathbf{B} \longrightarrow I_c(\mathbf{B})$*;* $f(x) = (x]$ *is a regular embedding.*

Proof. Clearly $I_c(\mathbf{B})$ is a bounded lattice. we show that it is also join-infinite distributive. Obviously $\bigvee^c_{J\in\Gamma} I \wedge J \leq I \wedge \bigvee^c \Gamma$, for every closed ideal I and every subset Γ of $I_c(B)$ with $\bigvee^c \Gamma = I_c(\cup\Gamma)$. Let x be an element of $I \wedge \bigvee^c \Gamma$, then $x \in I$ and $x = \bigvee\{y | y \leq x \text{ and } y \in J \text{ for some } J \in \Gamma\}$, by previous Proposition. Therefore $x \in I_c(\cup_{J\in\Gamma} I \wedge J) = \bigvee^c_{J\in\Gamma} I \wedge J$.

It is clear that for ideals I and J, $J \subseteq I \to^c J$. For ideals I, J and K we show that $I \vee^c J \to^c K = (I \to^c K) \wedge (J \to^c K)$.

Let $x \in (I \to^c K) \wedge (J \to^c K)$. By the previous proposition we have

$$\begin{aligned} x &= \bigvee\{y | \exists z \in I \to^* K \text{ with } y \leq z\} \\ &= \bigvee\{y' | \exists z' \in J \to^* K \text{ with } y' \leq z'\} \\ &= \bigvee\{t | \exists u \in I \vee J \to^* K \text{ with } t \leq u\}. \end{aligned}$$

Note that in the third equality we used both join-infinite distributivity and the fact that $z \wedge z' \in (I \to^* K) \wedge (J \to^* K) = I \vee J \to^* K$, since $z \in I \to^* K$

and $z' \in J \to^* K$. The converse inclusion is trivial. We can prove the other properties of " $\to$ " by similar arguments.

By Theorem 2, f is an embedding. Now we show that f preserves all infinite joins. Suppose that $\bigvee S$ exists for $S \subseteq B$. Let $U = \{f(s) \mid s \in S\}$, then $\bigvee S \in I_c(\bigcup U)$. Therefor $f(\bigvee S) = I_c(\bigcup U) = \bigvee^c U = \bigvee^c_{s\in S} f(s)$.

Similar to the ideal completion of basic algebras we have

Proposition 3. *The basic algebra $I_c(\mathbf{B})$ is a Heyting algebra if and only if $\mathbf{B}$ is a Heyting algebra.*

Proof. Note that for every closed ideal I, $(1] \to^c I = I$ if and only if $(1] \to^* I = I$. Now apply Proposition 1.

Proposition 4. *Every join-infinite distributive basic algebra can be regularly embedded in a complete Heyting algebra as a lattice.*

As we mentioned before, in the case of Heyting algebras the ideal completion and MacNeille completions coincide. Here we will generalize this fact for join-infinite distributive basic algebras. Let us recall the following definition

Definition 3. *Let $\mathbf{B}$ and $\mathbf{B}^*$ be join-infinite distributive basic algebras which $\mathbf{B}^*$ is also a regular completion of $\mathbf{B}$ with related embedding f. $\mathbf{B}^*$ is called a join dense completion of $\mathbf{B}$ if for any element a in $\mathbf{B}^*$ we have $a = \bigvee\{f(x) |\ f(x) \leq a \ \text{ and } \ x \in B\}$.*

Theorem 9. *Let $\mathbf{B}$ be a join-infinite distributive basic algebra. Then every join dense completion of $\mathbf{B}$ is lattice isomorphic to $I_c(\mathbf{B})$.*

Proof. Let f be the related embedding of $\mathbf{B}$ into $I_c(\mathbf{B})$ and $\mathbf{B}^*$ be a join dense completion of $\mathbf{B}$ with related embedding g. For ever closed ideal I we define $h(I) = \bigvee_{x\in I} g(x)$. We show that h is the required isomorphism. Let I and J be two closed ideals. Since $\mathbf{B}^*$ is join-infinite distributive we have $h(I) \wedge h(J) = \bigvee_{x\in I} g(x) \wedge \bigvee_{y\in J} g(y) = \bigvee_{x\in I} \bigvee_{y\in J} g(x \wedge y) \leq \bigvee_{z\in I\wedge J} g(z) = h(I \wedge J)$. The other direction is trivial. Now we show that h preserves infinite joins. Let Γ be a set of closed ideal, it is sufficient to show that $h(\bigvee^c \Gamma) \leq \bigvee_{I\in\Gamma} h(I)$. For $x \in \bigvee^c \Gamma$, $x = \bigvee\{y | y \leq x \text{ and } y \in \bigcup \Gamma\}$. Then $g(x) = \bigvee\{g(y) | y \leq x \text{ and } y \in \bigcup \Gamma\}$, since g is regular. Now for every $y \in \bigcup \Gamma$ there exists an ideal I in Γ such that $y \in I$, so $g(x) \leq \bigvee_{z\in I} g(z) = h(I)$. Now one can get the required inequality.

For injectivity suppose that $h(I) = h(J)$. Then $\bigvee_{x\in I} g(x) = \bigvee_{y\in J} g(y)$. So for every $x \in I$, $g(x) \leq \bigvee_{y\in J} g(y)$ which implies that $g(x) = \bigvee_{y\in J} g(x \wedge y)$. We show that $x = \bigvee_{y\in J} x \wedge y$ and so $x \in J$. Clearly x is a upper bound. Suppose that z is an element such that for any $y \in J$, $x \wedge y \leq z$. Then $g(x) = \bigvee_{y\in J} g(x \wedge y) \leq g(z)$, it implies that $x \leq z$, since g is injective. Hence, $I = J$.

For surjectivity of h suppose that $x \in \mathbf{B}^*$, then $x = \bigvee\{g(y) |\ g(y) \leq x \text{ and } y \in B\}$. It is easy to see that $h(\bigvee^c\{f(y) |\ g(y) \leq x \text{ and } y \in B\}) = x$, since h and g preserve infinite joins.

In fact the above proof shows that $h\ o\ f = g$. It means that our isomorphism is unique, since suppose that h' is another isomorphism with the same properties. Then for any closed ideal I we have $h(I) = h(\bigvee^c\{f(a)|\ a \in I\}) = \bigvee\{h(f(a))|\ a \in I\} = \bigvee\{h'(f(a))|\ a \in I\} = h'(I)$. As a corollary we have

Corollary 1. *Let* $\mathbf{B}$ *be a join-infinite distributive basic algebra and* I *be an ideal in* $\mathbf{B}$. *I is a closed ideal if and only if it is a normal ideal.*

Proof. First note that our related regular embedding in the MacNeille completions is $g(x) = (x]$. We show that the above unique isomorphism h is the identity function, i.e., for any closed ideal I, $h(I) = I$.

$h(I) = h(\bigvee^c\{f(x)|\ x \in I\}) = \bigvee^c\{h(f(x))|\ x \in I\} = \bigvee^c\{\bigvee_{y\leq x} g(y)|\ x \in I\} = \bigvee^c\{(x]|\ x \in I\} = I$. So every closed ideal is a normal ideal.

For the converse, suppose I is a normal ideal, $S \subseteq I$ and $\bigvee S = a$. If $x \in U(I)$, then $a \leq x$, and hence $a \in LU(I) = I$.

Corollary 2. *The class of all join-infinite distributive basic algebras is closed under MacNeille completions.*

4 Algebraic Completeness of Visser's Predicate Logic

Visser's basic predicate logic is a predicate logic with intuitionistic language which is interpreted in Kripke models with transitive accessibility relation. The propositional part of the logic was first introduced by Visser [9], called basic propositional logic, BPL, and developed by Ardeshir and Ruitenburg [5]. Basic predicate logic, BQL, as a predicate extension of BPL was first introduced by Ruitenburg in [8].

The language of BQL, $\mathcal{L}$, contains a denumerable set of predicate symbols of each finite arity, a denumerable set $\mathbf{V}$ of variable symbols, parentheses, logical constants $\top$ and $\bot$, the logical connectives $\wedge$, $\vee$, $\rightarrow$ and quantifiers $\exists$ and $\forall$. Our language is freed of function symbols. We usually include the binary predicate $=$ for equality. Formulas are defined as usual, except for the universally quantified formulas. A universally quantified formula is of the form $\forall\mathbf{x}(\phi(\mathbf{x}) \rightarrow \psi(\mathbf{x}))$, in which $\mathbf{x} = (x_1, x_2, \cdots, x_n)$, a finite sequence of distinct variables. A sequent is an expression of the form $\phi \Rightarrow \psi$, in which ϕ and ψ are formulas. We often write ϕ for $\top \Rightarrow \phi$. A rule with a double horizontal line means a two direction rule. For more details see [8]. We give an axiomatization with the axioms and rules

1) $\phi \Rightarrow \phi$, 2) $\bot \Rightarrow \phi$, 3) $\phi \Rightarrow \top$, 4) $\top \Rightarrow x = x$,

5) $\phi \wedge (\psi \vee \eta) \Rightarrow (\phi \wedge \psi) \vee (\phi \wedge \eta)$,

6) $\exists x\phi \wedge \psi \Rightarrow \exists x(\phi \wedge \psi)$, x is not free in ψ,

7) $x = y \wedge \phi \Rightarrow \phi[x/y]$, A is atomic, 8) $\forall\mathbf{x}(\phi \rightarrow \psi) \wedge \forall\mathbf{x}(\psi \rightarrow \eta) \Rightarrow \forall\mathbf{x}(\phi \rightarrow \eta)$,

9) $\forall \mathbf{x}(\phi \to \psi) \wedge \forall \mathbf{x}(\phi \to \eta) \Rightarrow \forall \mathbf{x}(\phi \to \psi \wedge \eta)$,

10) $\forall \mathbf{x}(\phi \to \eta) \wedge \forall \mathbf{x}(\psi \to \eta) \Rightarrow \forall \mathbf{x}(\phi \vee \psi \to \eta)$,

11) $\forall \mathbf{x}(\phi \to \psi) \Rightarrow \forall \mathbf{y}(\phi \to \psi)$, no variable in $\mathbf{y}$ is free on the left hand side,

12) $\forall \mathbf{x}(\phi \to \psi) \Rightarrow \forall \mathbf{x}(\phi[\mathbf{x}/\mathbf{t}] \to \psi[\mathbf{x}/\mathbf{t}])$, no variable in $\mathbf{t}$ is bounded in ϕ or ψ,

13) $\frac{\phi \Rightarrow \psi \quad \psi \Rightarrow \eta}{\phi \Rightarrow \psi}$, 14) $\frac{\phi \Rightarrow \psi \quad \phi \Rightarrow \eta}{\phi \Rightarrow \psi \wedge \eta}$, 16) $\frac{\phi \Rightarrow \eta \quad \psi \Rightarrow \eta}{\phi \vee \psi \Rightarrow \eta}$,

17) $\frac{\phi \wedge \psi \Rightarrow \eta}{\phi \Rightarrow \forall \mathbf{x}(\psi \to \eta)}$, no variable in $\mathbf{x}$ is free in ϕ,

18) $\frac{\phi \Rightarrow \psi}{\exists x \phi \Rightarrow \psi}$, x is not free in ψ,

19) $\frac{\phi \Rightarrow \psi}{\phi[\mathbf{x}/\mathbf{t}] \Rightarrow \psi[\mathbf{x}/\mathbf{t}]}$, no variable in $\mathbf{t}$ is bounded in the denominator,

20) $\forall \mathbf{y} x(\phi \to \psi) \Rightarrow \forall \mathbf{y}(\exists x \phi \to \psi)$, x is not free in ψ.

BQL without the last axiom above is shown by BQL^-. In the following, we prove the algebraic completeness and soundness for BQL^-.

An algebraic model is a structure $\underline{\mathbf{B}} = \langle \mathbf{B}, \mathbf{D}, I \rangle$, where $\mathbf{B}$ is a join- infinite distributive complete basic algebra and $\mathbf{D}$ is a non-empty set. Let $\bar{\mathbf{D}}$ be the set of names of all elements in $\mathbf{D}$. I is a map from atomic sentences of $\mathcal{L}(\mathbf{D})$, the language $\mathcal{L}$ expanded with the constants in $\overline{\mathbf{D}}$, to $\mathbf{B}$ which satisfies the following conditions: for any a and b in $\bar{\mathbf{D}}$, and for nay atomic formula $P(x)$,

$I(a = a) = 1$, and $I(P(a)) \wedge I((a = b)) \leq I(P(b))$.

The map I can be uniquely extended to the set of all sentences on $\mathcal{L}(\mathbf{D})$. For quantified formulas we have:

1. $I(\forall x(\phi \to \psi)) = \bigcap_{a \in \bar{\mathbf{D}}} I(\phi[x/a] \to \psi[x/a])$, and
2. $I(\exists x \phi) = \bigcup_{a \in \bar{\mathbf{D}}} I(\phi[x/a])$.

For $\phi, \psi \in For(\mathcal{L})$, suppose that $FV(\phi) \cup FV(\psi) = \{x_1, ..., x_n\}$. A sequent $\phi \Rightarrow \psi$ is satisfied in $\underline{\mathbf{B}}$, $\underline{\mathbf{B}} \models \phi \Rightarrow \psi$, if for any $a_1, ..., a_n \in \mathbf{D}$, $I(\phi[x_1/a_1, ..., x_n/a_n]) \leq I(\psi[x_1/a_1, ..., x_n/a_n])$. A sequent $\phi \Rightarrow \psi$ is valid, $\models \phi \Rightarrow \psi$, if for every model $\underline{\mathbf{B}}$, $\underline{\mathbf{B}} \models \phi \Rightarrow \psi$.

Theorem 10. *(***Soundness and completeness***) For any formulas ϕ and ψ, $BQL^- \vdash \phi \Rightarrow \psi$ if and only if $BQL^- \models \phi \Rightarrow \psi$.*

Proof. The proof of the soundness part of the theorem proceeds by induction on the height of the derivation. For the completeness part, we construct the Lindenbaum algebra $\mathcal{U}$ of BQL. Consider the equivalence relation $\sim$ defined in the set of all formulas by: $\phi \sim \psi$ iff $\vdash \phi \Rightarrow \psi$ and $\vdash \psi \Rightarrow \phi$. This relation is a congruence with respect to the operations associated with the connectives. Then $\mathbf{U}$ is the quotient algebra of the algebra of formulas given by the operations

associated with the connectives, $\top$ and $\bot$. One may easily check that the algebra $\mathbf{U}$ is a basic algebra whose lattice order is such that $[\phi] \leq [\psi]$ iff $\vdash \phi \Rightarrow \psi$. In addition we have:

$$[\exists x\phi] = \bigcup_{y \in V} [\phi[x/y]]; \qquad [\forall x(\phi \to \psi)] = \bigcap_{y \in V} [\phi[x/y] \to \psi[x/y]].$$

We check the first equality. It is easy to show that $\vdash \phi(x) \Rightarrow \exists x\phi(x)$. So, $\vdash \phi[x/y] \Rightarrow \exists x\phi(x)$, for every variable y. Therefore $[\exists x\phi]$ is an upper bound for the set $\{[\phi[x/y]] | y \in V\}$. To show that it is the least upper bound suppose that ψ is a formula, such that $\vdash \phi[x/y] \Rightarrow \psi$ for all $y \in \mathbf{V}$. If $x \notin FV(\psi)$, then take x for y. So we have $\vdash \phi(x) \Rightarrow \psi$, then we can deduce $\vdash \exists x\phi(x) \Rightarrow \psi$, which implies $[\exists x\phi] \leq [\psi]$. If $x \in FV(\psi)$, take fresh variable z. Then, by our assumption, we have $\vdash \phi[x/z] \Rightarrow \psi$. Therefore $\vdash \exists z\phi[x/z] \Rightarrow \psi$, which implies $[\exists z\phi[x/z]] \leq [\psi]$. By axiom (6), the basic algebra $\mathbf{U}$ is countable join-infinite distributive basic algebra but it does not have to be complete. By Theorem 8, $\mathbf{U}$ can be embedded into a complete basic algebra $\mathbf{B}$, in which the infinite meets and infinite joins of elements of $\mathcal{U}$ are preserved. Note that in this case the complete algebra $\mathbf{B}$ is also countable. Consider a model $\underline{\mathbf{B}} = \langle \mathbf{B}, \mathbf{V}, \mathbf{I} \rangle$ where $\mathbf{V}$ is the set of variables and $\mathbf{I}$ is the map from the atomic sentences of $\mathcal{L}(\overline{\mathbf{V}})$ given by $\mathbf{I}(\phi(\overline{x_1}, \ldots, \overline{x_n})) = [\phi(x_1, \ldots, x_n)]$ for every atomic formula $\phi(x_1, \ldots, x_n)$. By induction it follows that $\mathbf{I}(\phi(\overline{x_1}, \ldots, \overline{x_n})) = [\phi(x_1, \ldots, x_n)]$ for every formula $\phi(x_1, \ldots, x_n)$. Now we are in a position to prove the completeness. Suppose $\models \phi \Rightarrow \psi$. Then for any model $\underline{\mathbf{C}}$, $\underline{\mathbf{C}} \models \phi \Rightarrow \psi$. Take $\underline{\mathbf{B}}$ for $\underline{\mathbf{C}}$, then $[\phi] \leq [\psi]$, which means $\vdash \phi \Rightarrow \psi$.

By the same method as above we can prove one direction of the above theorem for BQL.

Theorem 11. *(***Completeness***) For any formulas ϕ and ψ, if $BQL \models \phi \Rightarrow \psi$, then $BQL \vdash \phi \Rightarrow \psi$.*

To prove the soundness for BQL we need to prove that the regular completions of join-infinite basic algebras satisfy clause 5 of Theorem 3. At this moment, unfortunately, we don't know if the closed ideal completion satisfies this property or not. It is an interesting problem to introduce a complete algebraic semantics for BQL.

References

1. Alizadeh, M., Ardeshir, M.: Amalgamation property for the class of basic algebras and some of its natural subclasses. Archive for Mathematical Logic 45, 913–930 (2006)
2. Alizadeh, M., Ardeshir, M.: On Löb algebras. Mathematical Logic Quarterly 52, 95–105 (2006)
3. Alizadeh, M., Ardeshir, M.: On Löb algebras II (submitted)
4. Ardeshir, M.: Aspects of basic logic. PhD thesis, Department of Mathematics, Statistics and Computer Science, Marquette University (1995)

5. Ardeshir, M., Ruitenburg, M.: Basic propositional calculus I. Mathematical Logic Quarterly 44, 317–343 (1998)
6. Harding, J.: Any lattice can be regularly embedded into the Macneille completion of a distributive lattice. Houston J. Math. 19, 39–44 (1993)
7. Harding, J., Bezhanishvili, G.: MacNielle completions of Heyting algebras. Houston J. Math. 30, 937–950 (2004)
8. Ruitenburg, W.: Basic predicate calculus. Notre Dame Journal of Formal Logic 39, 18–46 (1998)
9. Visser, V.: A propositional logic with explicit fixed points. Studia Logica 40, 155–175 (1981)

Transformations via Geometric Perspective Techniques Augmented with Cycles Normalization

Gleifer V. Alves, Anjolina G. de Oliveira, and Ruy de Queiroz

Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
{gva,ago,ruy}@cin.ufpe.br

Abstract. A normalization procedure is presented for a classical natural deduction (ND) proof system. This proof system, called N-Graphs, has a multiple conclusion proof structure where cycles are allowed. With this, we have developed a thorough treatment of cycles, including cycles normalization via an algorithm. We also demonstrate the usefulness of the graphical framework of N-Graphs, where derivations are seen as digraphs. We use geometric perspective techniques to establish the normalization mechanism, thus giving a direct normalization proof.

Keywords: proof theory, normalization, proof-graphs, multiple conclusion, cycles.

1 Introduction

Our main goal is to use geometric perspective techniques in order to establish a normalization procedure capable of handling derivations with multiple conclusion proof structure where cycles are admissible. Normalization is defined for N-Graphs, which has been conceived by de Oliveira in her doctoral thesis ([1], [2]), as a suitable solution to the lack of symmetry in classical ND logic. The resulting proof system defined by de Oliveira is a multiple conclusion natural deduction system augmented with structural rules. The structural rules together with constants ($\bot$, $\top$) have a key role in building proof-graphs. Derivations in N-Graphs are drawn as digraphs. The multiple conclusion proof structure allows for cycles in proof-graphs.

Now we briefly describe the two scenarios which correspond to our efforts: geometric perspective and normalization for classical ND logic. The term 'geometric perspective' is used to denote a particular way of abstracting from syntactic manipulation by turning explicit certain implicit symmetries in proof calculi via the extraction of formula occurrence flow graphs. With this we define an abstract graphical framework, as an abstract representation of *proofs*, *proof objects* (*i.e.*, formulas, rules, operators, etc) and related *properties* (*e.g.*, soundness and normalization) by means of graphical resources, where these resources are given via graph-theoretic concepts and mechanisms.

There are different kinds of frameworks. Here we should mention the graph framework used in Statman's pioneering work [3]. Besides, the logical flow graphs

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 84–98, 2009.

framework of S. Buss are used to trace the flow of formula-occurrences. For instance, we cite Carbone's work [4], where flow graphs are used to analyze the cut-elimination procedure. Also N-Graphs and Alessio–Gundersen work [5] bring the use of flow graphs. Yet we mention the circuit framework, which is used in the work of Blue et. al. [6], where proof-nets are drawn as *circuits*.

Our normalization procedure is devised for classical ND, where the work of Prawitz appears as a seminal reference [7]. Moreover, we mention some extensions of Prawitz normalization, Statman, Massi-Pereira [8] and Stålmarck [9] works, in order to define normalization for the full set of ND operators. These works are for single conclusion systems. But we are also interested on normalization for multiple conclusion systems. As established in the works of Ungar [10], Cellucci [11] and Blute et. al. The first two authors define normalization without using a graphical framework, while Blute et. al. use a graphical representation for proof-nets.

We realize that on the one hand, some works are devised to define normalization guided by an abstract graphical framework. On the other hand, some works use a framework to represent the system and it's rules, but do not represent transformations among derivations. Or yet the normalization does not use graphical representations. Now we remark those works which define normalization by means of some abstract graphical framework. Wwe cite Statman's work, where the *homomorphism* concept is used to establish a correspondence among proof systems. We also have the work of Blute et. al., where the graphical framework of proof-nets is used to describe the reduction rules and cycles. But cycle structures are not normalized. Yet we mention Carbone's work where flow graphs are used to analyze the process of cut-elimination in classical sequent calculus. Additionally the work of Alessio–Gundersen shows the use of an abstract graphical framework to represent a proof system and the normalization procedure. Notice Alessio–Gundersen admit and handle cycle structures. But they do not normalize cycles.

We notice that only the first two works aforementioned define normalization for classical ND logic. Besides, only Blute et. al. and Alessio–Gundersen admit cycles. However, they both do not present a normalization for cycles. As a result, there is a lack of a specific mechanism capable to check whether or not a given (*valid*) cycle has a detour.

2 Related Work

In this section we briefly present those works which are devised to define a normalization mechanism by means of some abstract graphical framework.

Firstly Statman has used the concept of *homomorphism* in order to determine the correspondence among two different proof systems: X and Y. With such correspondence it is possible to indirectly define normalization for X, since normalization for Y is already known. Secondly we show Carbone's work, where logical flow graphs have been used to analyze the complexity of proofs in sequent calculus. Logical flow graphs are constructed by tracing the flow of information

(*i.e.*, the formulas) in proofs of the sequent calculus. In logical flow graphs the different occurrences of a formula in a proof are linked by the edges of the graph. Thirdly we mention the work of Blute et. al. [6], where the authors intend to define translations among proof-nets and categories. The circuit framework is used to represent the proof-nets. And the normalization mechanism is guided by an abstract graphical framework.

In [5], Alessio–Gundersen present normalization via an abstract graphical framework for SKS system. SKS is a formalism for the calculus of structures in deep inference logic. The abstract framework is defined via the atomic flows, which are directed graphs obtained from a derivation by only retaining information about the creation and destruction of atom occurrences. Atomic flows are based on S. Buss *logical flow graphs*. The abstract framework has a simple representation of derivations, since logical rules are discharged and only structural rules are used. Cycle structures are represented in atomic flows. However cycles are only handled in one fragment of normalization reductions. When all normalization reductions are used, cycles should be dismissed. Otherwise normalization would be non-terminating.

3 N-Graphs

N-Graphs has logical and structural rules. Derivations are graphically represented by means of *labelled digraphs*, augmented with a multiple conclusion proof structure. Thus introduction and elimination rules share a symmetry. And cycle structures are allowed in derivations. Proofs are represented by *proof-graphs*, which are graphs whose *vertices* are labelled with formula occurrences and *edges* represent atomic steps in a derivation. And N-Graphs are those proof-graphs which are logically sound. The propositional language of N-Graphs has the following elements: (**i**) *propositional constants*: $\bot, \top$; (**ii**) *connectives*: $\neg, \wedge, \vee, \rightarrow$; (**iii**) *formula-occurrences*: $A, B, C, \ldots$. And the grammar of the language is given as follows, where p ranges over a countable set of propositional letters: $A, B := p \mid \top \mid \bot \mid \neg A \mid A \wedge B \mid A \vee B \mid A \rightarrow B$.

Definition 1 ((Focussing/Defocussing) Branch point). *A* **branch point** *is a vertex in a digraph with three edges attached to it; while a* **focussing branch point** *is a vertex in a digraph with two edges oriented towards it; and a* **defocussing branch point** *is a vertex in a digraph with two edges oriented away from it.*

Definition 2. (**i**) A **focussing link** is a set $\{(u_1, v), (u_2, v)\}$ in which v is called the *branch point* (or focussing branch point) of this link. u_1 and u_2 are the premises and v is the conclusion. (**ii**) A **defocussing link** is a set $\{(u, v_1), (u, v_2)\}$ in which u is called the *branch point* (or defocussing branch point) of this link. v_1 and v_2 are the conclusions and u is the premiss. (**iii**) A **simple link** is an edge (u, v) which belongs neither to a focussing nor to a defocussing link. u is the premiss and v is the conclusion.

Definition 3 (Proof-graph). *A proof-graph is a connected directed graph G defined as follows: (***i***) G is constructed by means of three kinds of links: simple, focussing and defocussing.* N-Graphs *links are divided into logical and structural links; (***ii***) there are two kinds of edges: meta and solid. Meta-edges are labelled with m in order to identify the respective edge, where a meta-edge is used to represent cancellation of a hypothesis. Solid edges are left unlabelled; (***iii***) each vertex is labelled with a formula-occurrence; (***iv***) every vertex in a proof-graph is labelled with a conclusion of a unique link and is the premiss of at most one link.*

In Fig. 1 we present the logical and structural links of N-Graphs. Notice expansion and contraction links are also named as switching links. And their edges are respectively called switching edges.

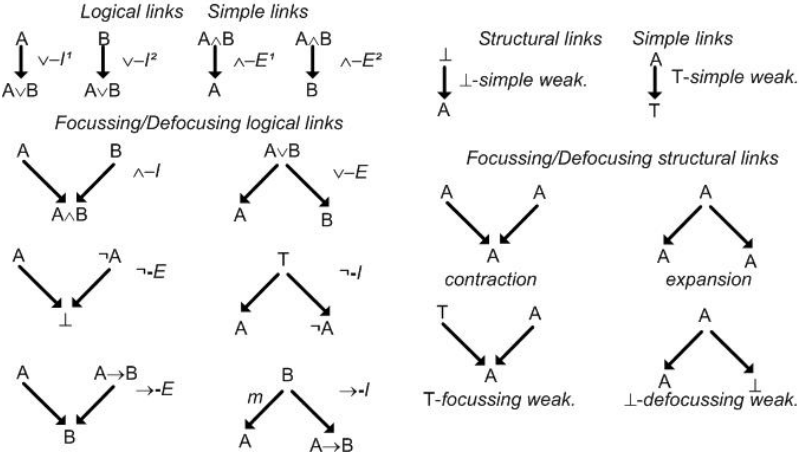


Fig. 1. N-Graphs links

Definition 4 (Conjunctive and disjunctive links). *The links ∧-I, ¬-E, → -E, ⊤-focussing-weakening and expansion link are called* **conjunctive**. *While the links ∨-E, ¬-I, → -I, ⊥-defocussing-weakening and contraction link are called* **disjunctive**.

3.1 Cycles in N-Graphs

In a proof-graph we may have multiple formulas as premisses and conclusion nodes. The remaining formulas (which are neither premisses nor conclusions) are called nodes. When the proof-graph is a cycle we extend these notions. So, respectively, we have: cycle premiss node (or CP), cycle conclusion node (or CC) and those nodes in a cycle which are neither CP nor CC are the cycle nodes (or CN). Every logical link has two sorts of formulas: cutting center and cutting periphery. Cutting center is the formula which has the connective that is being

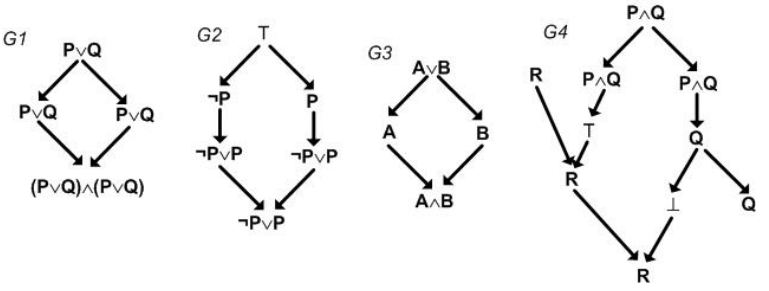


Fig. 2. Cycle structures: examples

introduced or eliminated. Cutting periphery is the formula which neither has the connective that is being introduced nor eliminated. For example, in links $\neg$-I and $\neg$-E, A is a cutting periphery and $\neg A$ is a cutting center. This terminology is suggested by Statman in his thesis [3].

De Oliveira has defined the soundness criterion for N-Graphs based on Danos-Regnier criterion for proof-nets [1]. The switching links (expansion and contraction) are *necessarily* used to build cycles in N-Graphs. The criterion chiefly states the following: one of the two edges of each *switching link* should be removed. By removing these edges one must check whether or not the resulting graph is *acyclic* and *connected*, in this case the proof-graph in an N-Graph, *i.e.*, a sound proof.

With this, we can extract the two basic patterns of simple valid cycles: (**i**) The CP is premiss of an *expansion* link and the CC is conclusion of a *focussing* and *conjunctive* link; (**ii**) The CP is premiss of a *defocussing* and *disjunctive* link and the CC is conclusion of a *contraction* link. In Fig. 2 we bring examples. Notice that G_1, G_2 and G_4 are valid cycles. While G_3 is a non-valid cycle, since G_3 has no switching link at all.

4 Normalization for N-Graphs

In this section we present a set of cut rules responsible to remove three kinds of detours from N-Graphs. The first one is defined bellow. Before state this definition we mention that an I-flavour element represents both: a logical introduction link and a $\bot$ link where it's conclusion is a non-atomic node, while an E-flavour element represents both: a logical elimination link and a $\top$ link where it's premiss is a non-atomic node.

Definition 5 (cut node). *A proof-graph G has a* **cut node** α *when α is a cutting center of an* I-flavour *element and at the same time it is a cutting center of an* E-flavour *element. Notice* I-flavour *and* E-flavour *should share the same connective.*

In order to remove the **cut nodes** we define β cut transformations, **logical**-β and **structural**-β. A **logical**-β is defined when there is logical introduction link followed by an elimination link. In Fig. 3 we present at left a **logical**-β.

The second kind of detour is named **cut path**.

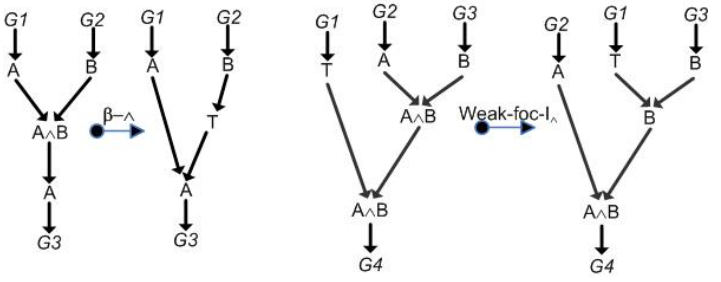


Fig. 3. Cuts: β and permutative weakening

Definition 6 (cut path). *A proof-graph G which has a* **cut path** *is defined as follows: (1). G has an* I-flavour *element where ϕ is the respective cutting center node; (2). Next G has a sequence of one or more weakening structural links where ϕ is propagated via these structural links; (3). And G has an* E-flavour *element where ϕ is the respective cutting center node. Thus, a* **cut path** *is established by the sequence of ϕ nodes existent in G. Notice that a cut path may have the three parts defined above, alternatively the first or the last part may be empty.*

The so-called **permutative weakening cut transformations** are conceived to remove the cut paths. The basic mechanism used by all weakening transformations is simple and can be summarized as follows: I-flavour elements should be pushed bellow the weakening structural links, while the E-flavour elements should be pushed above the weakening structural links. At right of Fig. 3 one permutative weakening cut is shown. The third detour is called **cut hole**.

Definition 7 (cut hole). *A cut hole is a subgraph H from a proof-graph G (as illustrated at left of Fig. 4), where the following holds: Subgraph H may have a single formula or n links, where any link is admissible, i.e., simple, focussing, defocussing, logical and structural links; Δ is the set of premisses of H; Γ is the set of conclusions of H; Δ and Γ should have only $\bot$ and/or $\top$ constants; Proviso: Δ must have at least one $\bot$ constant.*

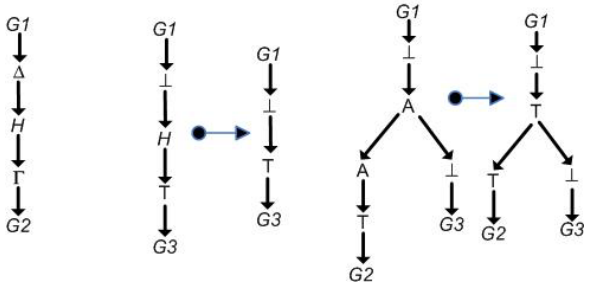


Fig. 4. $\bot$:$\top$ cuts

Roughly speaking when we have some $\bot$ constant and we simply want to obtain some $\top$ constant, we understand that these constants should be directly obtained via $\bot$ link and/or structural links. As a result, any other link (or node) placed among these constants is placed in a hole and should be removed. We define two kinds of $\bot$:$\top$ cut transformations to remove **cut holes**. The first kind is the simple $\bot$:$\top$ cut which is defined when H has a single node or only simple links. The second kind is the extended $\bot$:$\top$ cut and it is defined when H has at least one focussing or defocussing link. Fig. 4 shows both kinds of $\bot$:$\top$ cuts.

5 Cycles in Proof-Graphs

N-Graphs normalization demands a treatment of cycles which includes a definition of (valid) classes and a specific algorithm.

5.1 Classes of Cycles

At subsection 3.1 we had defined the terms: CP, CC and CN for those nodes which are placed in a cycle. Now we extend these definitions. Previously on Fig. 2 we have seen three (sound) cycles. Note that G_4 cycle has some branches. Thus, we have: branch cycle premiss node (BP) and branch cycle conclusion node (BC). Specifically G_4 has node R as BP and Q as BC. Yet cycles have the so-called initial and final links. An initial link is a defocussing link, while a final link is a focussing link. G_1 has *expansion* as initial and $\wedge$-I as final link.

There are two classes of N-Graphs cycles, the **basic** and the **recursive** cycles which are builded from the basic class.

Definition 8 (basic cycle). *A basic cycle has one* CP *and one* CC*, the* CP *belongs to the initial link and the* CC *to the final link. Moreover, it has two parallel structures connecting these two links. The* left subgraph *(or* LS*) and* right subgraph *(or* RS*). The subgraphs can be empty, or may have any kind of link. Thus, three different structures for subgraphs can be obtained: (1). a* **diamond** *structure: when both subgraphs (*LS *and* RS*) are empty, in Fig. 2 G_1 is a* **basic diamond** *cycle; (2). a* **unbranched** *structure: when at least one of the subgraphs (*LS *and* RS*) has only simple links, in Fig. 2 G_2 is a* **basic unbranched** *cycle; (3). a* **branched** *structure: when at least one of the subgraphs (*LS *and* RS*) necessarily has focussing and/or defocussing links, in Fig. 2 G_4 is a* **basic branched** *cycle. Notice that in order to obtain the three basic cycles we still must apply the soundness criterion.*

The class of **recursive cycles** is divided into two kinds of cycles.

Definition 9 (recursive cycle). *The* **recursive sequence cycle** *is defined by a sequence of cycles. It has y_n elements $(y_1, \ldots, y_n,\ n \geq 2)$, where some* CC *of y_1 is connected to some* CP *of y_2, and so on. An example is shown at left of Fig. 5. The* **recursive nested cycle** *has a structure with initial and final links which are connected via two parallel structures named* left nested *(*LN*) and* right nested *(*LN*). These parallel structures admit any kind of cycle. At right of Fig. 5 an example is given.*

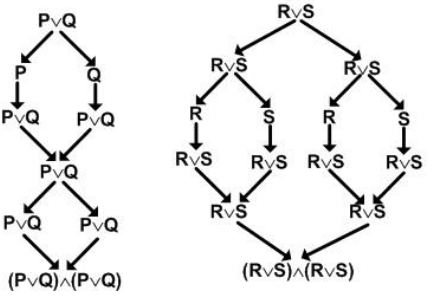


Fig. 5. Examples: sequence and nested cycles

The soundness criterion for recursive cycles says that each cycle must be independently checked for soundness. Additionally the nested cycle should have exactly the initial or the final link as a *switching* link.

5.2 Cycle Algorithm

To normalize N-Graphs cycles we have defined an algorithm named cut-cycle checking algorithm (or 3CA). 3CA is responsible to determine whether or not a given cycle is redundant (*i.e.*, a **cut cycle**). The algorithm works as follows: given a proof-graph G; we obtain the same proof-graph G, in case G is not a cut cycle; or we obtain a new proof-graph G', in case G is a cut cycle. The basic mechanism of 3CA is defined to determine if it's possible to obtain a tree from some premisses towards some conclusions of G, if so, then G is transformed in G'. Fig. 6 shows an example of 3CA application, where the following terms are used: LP/LC for lists of premisses/conclusions; LN for list of matched nodes; and LRP/LRC for lists of remaining premisses/conclusions. The cycle (shown at Fig. 6) is a cut cycle, since a tree from the cycle structure has been obtained by means of an $\rightarrow$-E link (see center of Fig. 6). Next the resulting tree is obtained by joining at the aforementioned tree the remaining node R (see right of Fig. 6).

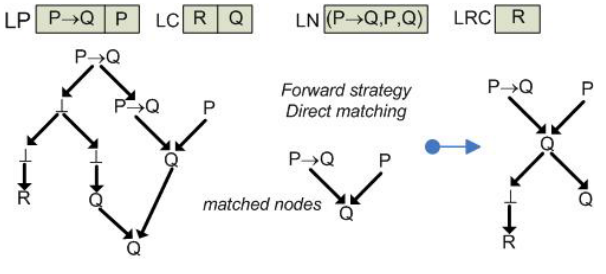


Fig. 6. 3CA - example

6 Normalization Proof

We shall demonstrate that the set of cut transformations together with the 3CA algorithm are responsible to transform a given redundant N-Graph into a non-redundant N-Graph. In order to build the proof we give some definitions, property and trace an analysis of the reduction strategies.

Proposition 1 (Soundness preservation). *Let G be an* N-Graph*, and G' a proof-graph resulting from the application of a cut transformation to G, then G' is also an* N-Graph*.*

Proof idea. By inspection over the sets of cut transformations and the 3CA algorithm. □

Definition 10 (Cut/Cut free N-Graph). *A given proof-graph G is said to be a cut* N-Graph *when G has at least one of the following cut structures: cut node, cut path, cut hole or cut cycle. Conversely, if G does not any of the aforementioned cut structures, then G is a cut free* N-Graph*.*

There are three kinds of normalization reduction strategies: direct, hidden and overlapped reduction strategy. Before describing them we give the following definition.

Definition 11 (Valid reduction strategy). *A valid reduction strategy in a proof-graph G is determined by: (***i***) choosing the maximum cut structure C of G; (***ii***) applying the respective transformation of C; (***iii***) and check if the* normalization measure *of G has been decreased. We repeat this process until the* normalization measure *is reduced to zero and G becomes a cut free* N-Graph*.*

Note that each cut structure has a specific value assigned to it, this value is determined by the so-called normalization measure, which is defined in the Appendix.

The direct reduction strategy is used when we have a redundant N-Graph G with one or more cut structures. Then we simply choose the cut structure with the maximum cut structure and apply the corresponding transformation. The hidden reduction strategy is similar with the above strategy, but it has a major difference, the notion of a hidden cut.

Definition 12 (Hidden cut). *Given a proof-graph G with a cut structure C and considering we apply the corresponding transformation of C. We obtain a proof-graph G' which is free of C. Nonetheless, a new and unexpected cut structure C' has been uncovered at G'. Thus, we say C' is a hidden cut of C.*

The overlapped reduction strategy is determined by the following concepts.

Definition 13 (Overlapped cuts). *Given a proof-graph G, which has two cut structures: C_1 and C_2. If C_1 and C_2 have at least one node or link in common, then we say C_1 and C_2 are overlapped cuts.*

The overlapped strategy may lead to different strategies for a given proof-graph, or as defined bellow to a *critical pair*. When a given proof-graph has a critical pair one needs to determine if the different reduction strategies lead to a *convergent* or a *divergent* critical pair. For divergent critical pairs we need to build equivalence relations in order to obtain a *weak confluence* property.

Definition 14 (Critical pair). *If we have a proof-graph G with two overlapped cuts: C_1 and C_2, where we can draw two different reduction strategies: one starting at C_1 and other at C_2. Then, we say that C_1 and C_2 conceive a critical pair.*

Generally speaking, our **normalization proof** must demonstrate that every (valid) reduction strategy constructed in N-Graphs normalization reduces a cut N-Graph into a cut free N-Graph, is finite, weak confluent and has it's normalization measure decreased in each reduction step. The normalization proof together with related definitions and auxiliary examples are described in the Appendix.

7 Conclusion

Our normalization procedure for N-Graphs brings the following contributions: (**i**) normalization for the full set of N-Graphs operators; (**ii**) reductions for $\bot/\top$ operators; (**iii**) use of abstract graphical framework; (**iv**) treatment of cycle structures (*i.e*, classification and normalization); and (**v**) construction of a direct normalization proof. As forthcoming developments we intend to use normalization techniques to guide the investigation towards an identity of proofs for N-Graphs. Furthermore, we foresee the use of N-Graphs in the extension of Abramsky's proofs as processes paradigm from linear to classical logic [12].

References

1. de Oliveira, A.G.: Proofs from a Geometric Perspective. PhD thesis, Centro de Informática - Universidade Federal de Pernambuco (April (2001))
2. de Oliveira, A.G., de Queiroz, R.J.G.B.: 1 of Logic for Concurrency and Synchronisation. Trends in Logic - Studia Logic Library. In: Geometry of Deductions via Graphs of Proofs, pp. 1–86. Kluwer Academic Publishers, Dordrecht (2003)
3. Statman, R.: Structural Complexity of Proofs. PhD thesis, Stanford University (May 1974)
4. Carbone, A.: Duplication of directed graphs and exponential blow up of proofs. Ann. Pure Appl. Logic 100(1-3), 1–67 (1999)
5. Guglielmi, A., Gundersen, T.: Normalisation control in deep inference via atomic flows. Logical Methods in Computer Science 4, 1–36 (2008)
6. Blute, R., Cockett, J.R.B., Seely, R.A.G., Trimble, T.H.: Natural deduction and coherence for weakly distributive categories. Journal of Pure and Applied Algebra 13(3), 229–296 (1996)
7. Prawitz, D.: Ideas and results in proof theory. In: Fenstad, J.E., ed.: Proceedings 2nd Scandinavian Logic Symp., Oslo, Norway, 18–20 June (1970); Studies in Logic and the Foundations of Mathematics, vol. 63, pp. 235–307. North-Holland, Amsterdam (1971)

8. Pereira, L.C.P.D., Massi, C.D.B.: Normalização para a lógica clássica. III Encontro Nacional de Filosofia, 5 Gramado, RS - Brasil (1988)
9. Stålmarck, G.: Normalization theorems for full first order classical natural deduction. Journal of Symbolic Logic 56(1), 129–149 (1991)
10. Ungar, A.M.: Normalization, Cut-Elimination and the Theory of Proofs. CSLI Lecture Notes, vol. 28. CSLI Publications, Stanford (1992)
11. Cellucci, C.: Existential instantiation and normalization in sequent natural deduction. Annals of Pure and Applied Logic 58(2), 111–148 (1992)
12. Abramsky, S.: Proofs as processes. Theoretical Computer Science 135, 5–9 (1994)

Appendix: Normalization Proof

This appendix describes the normalization proof. In order to construct the proof we shall define the so-called normalization measure. Besides we illustrate some reduction strategies by means of examples.

Definition 15 (Cut structure degree). *The degree for each cut structure is given as follows, where $d(C)$ represents the cut structure degree*[1] *of a given cut C.*

- **cut node***: the cut node degree is determined by the formula degree which labels the cut node and it is represented as: $d(C) = d(\text{cut node})$.*
- **cut path***: the cut path degree is given by the degree from the node which is the cutting center of the introduction link or the cutting center of the elimination link and belongs to the cut path. This cut structure degree is represented by: $d(C) = d(\text{cutting center node})$. At left of Fig. 7 we have a cut path which has it's cutting center node given by $d(P \wedge Q)$.*
- **cut hole***: the degree of the cut hole is zero, it is represented by: $d(C) = 0$.*
- **cut cycle***: the cut cycle degree is established by the sum from the degrees of the following sets: cycle premiss (CP), branch cycle premiss (BP), cycle conclusion (CC) and branch cycle conclusion (BC) plus 1 as a penalty (considering the cycle as a redundant framework). Notice the sets BP and BC may be empty. The cut cycle degree is represented as follows: $d(C) = d(CP) + d(BP) + d(CC) + d(BC) + 1$. At center of Fig. 7 we have a cut cycle, where it's degree is determined by: $d(R) + d(P \wedge Q) + d(R) + d(Q) + 1$.*

Definition 16 (Amount of cut structure nodes). *Notation $n(C)$ represents the amount of nodes for a given cut structure C.*

- **cut node***: a cut node has a single node, $n(C) = 1$.*
- **cut path***: every cut path has a sequence of weakening structural links which starts at the cutting center node of an introduction link and/or ends at the cutting center node of an elimination link. With this, the cutting center node is propagated through the weakening structural links. Therefore, the amount of nodes in a cut path is given by the number of cutting center node copies which exist in the structural part. The amount is given by: $n(C) = s$, where $s \geq 2$. In the example at left of Fig. 7 the amount of nodes is equal to 3.*

[1] The **degree** of a formula A is defined as the number of occurrences of logical constants in A, except $\bot$ and $\top$ constants.

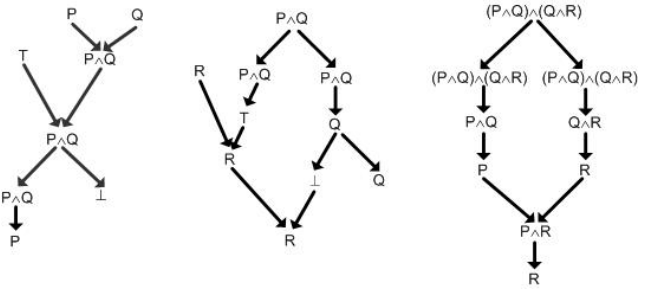


Fig. 7. Examples: cut path, cut cycle and cut periphery

- **cut hole***: the amount of nodes in a cut hole is established by the number of nodes which are placed inside the cut hole structure,* $n(C) = h$*, where* $h \geq 1$*.*
- **cut cycle***: the amount of nodes in a cut cycle is defined by the number of elements from the following sets:* CN*,* CP*,* BP*,* CC *and* BC*. Thus, we have:* $n(C) = n(CN) + n(CP) + n(BP) + n(CC) + n(BC)$*. Respectively, in the example at center of Fig. 7 the amount of nodes is:* $6 + 1 + 1 + 1 + 1$*.*

The third measure element is a specific case defined for cut nodes and cut paths which are placed in a cycle structure. We remark this cycle structure can not be a cut cycle. This measure is called cut periphery.

Definition 17 (Cut periphery). *Given a proof-graph* G *which has a cut node (or a cut path)* C*. If* C *is placed in a cycle structure* P*, where* P *is not a cut cycle, then, we say that* P *is the cut periphery of* C*, represented as:* $P(C) = 1$*.*

At right of Fig. 7 we shown an example with a cut periphery. Notice the cycle is not a cut cycle, but we do have a cut node at $P \wedge R$. As a result, the cycle is the cut periphery of $P \wedge R$.

In terms of the above concepts we determine our normalization measure.

Definition 18 (Normalization measure). *The* normalization measure *for a given proof-graph* G*, which has a maximum cut structure* C *is determined by the following tuple. We remark that if* G *has two (or more) maximum cut structures, for example,* C_1 *and* C_2*, then we can freely select* C_1 *or* C_2*.* normalization measure$(G) := \langle P(C), MaxCut(C), NCuts(G) \rangle$

- $P(C)$*: is the cut periphery of* C*, according with Def. 17.*
- $MaxCut(C)$*: is determined by the following pair:* $MaxCut(C) \doteq (d(C), n(C))$
  - $d(C)$*: is the degree of* C*, according with Def. 15.*
  - $n(C)$*: is the amount of nodes in* C*, according with Def. 16.*
- $NCuts(G)$*: is the total number of cut structures in* G*.*

We consider that a proof-graph G has been transformed into a new proof-graph G', by means of applying some cut transformations. Thus, we establish that normalization measure(G') should be less than normalization measure(G), respecting the following lexicographic order.

$$P(C') < P(C)$$
$$\text{or } P(C') = P(C) \text{ and } d(C') < d(C)$$
$$\text{or } P(C') = P(C) \text{ and } d(C') = d(C) \text{ and } n(C') < n(C)$$
$$\text{or } P(C') = P(C) \text{ and } d(C') = d(C) \text{ and } n(C') = n(C) \text{ and}$$
$$NCuts(G') < NCuts(G)$$

Before state the normalization proof we give further definitions related with overlapped reduction strategy.

Definition 19 ((Major/Minor) overlapped cut). *For a given proof-graph G with two overlapped cuts: C_1 and C_2. If we apply the reduction strategy of C_1 and as a side-effect C_2 is removed from G. Then, we say C_1 is the* **major** *overlapped cut of G. Conversely if we apply the reduction strategy of C_2 and C_1 remains at G. Then, we say C_2 is the* **minor** *overlapped cut of G.*

Definition 20 ((Different/Equivalent) overlapped cuts). *Given a proof-graph G with two overlapped cuts: C_1 and C_2, where C_1 is a major overlapped cut and C_2 is a minor overlapped cut, or vice-versa. We say G has* **different** *overlapped cuts. Conversely if the normalization measure of C_1 is equal to the normalization measure of C_2 and C_1 and C_2 are not defined as major/minor overlapped cuts. Then, we say G has* **equivalent** *overlapped cuts.*

Next we illustrate hidden and overlapped strategies via two examples. In Fig. 8 we present a hidden cut. Note that at left we have the cut N-Graph which has a cut node at $P \wedge R$. But, we realize that this cut node is placed in a cycle structure. As a result, this cut node has a *cut periphery*. Thus, we obtain nz measure:= $\langle 1, (1, 1), 1\rangle$. Next, we remove the cut node and a hidden cut arises in proof-graph, *i.e.*, a cut cycle. Therefore, the new measure is given by: $\langle 0, (4, 9), 1\rangle$. At last, the 3CA algorithm is applied and we obtain a cut free N-Graph.

Next Fig. 9 shows an overlapped cut among a *cut hole* and a *cut cycle*. We remark they are different overlapped cuts, since cut hole is a *major* overlapped cut and cut cycle is a *minor* overlapped cut. Yet, cut hole has nz measure:= $\langle 0, (0, 5), 2\rangle$ and cut cycle has nz measure:= $\langle 0, (1, 5), 2\rangle$. At left of Fig. 9 we present the first reduction strategy, where the cut hole is removed and the cut free N-Graph is directly obtained. At right of Fig. 9, we bring the second reduction

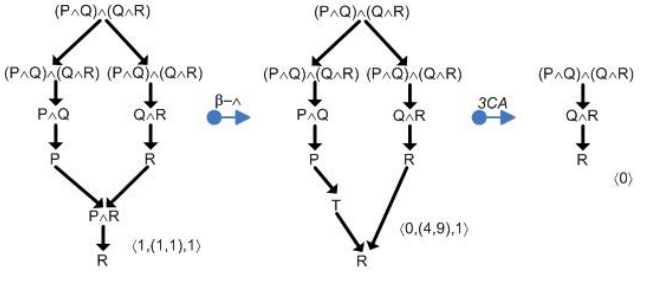


Fig. 8. Normalization proof - hidden cut example

strategy, where we have nz measure:= $\langle 0, (1, 5), 2\rangle$. Then, the cut cycle is removed and the cut hole is changed, but it remains in the proof-graph. Thus, the new nz measure:= $\langle 0, (0, 1), 1\rangle$. With this, the last step of reduction is responsible to remove the cut hole and obtain a cut free N-Graph. We still emphasize that this example yields a *convergent critical pair*.

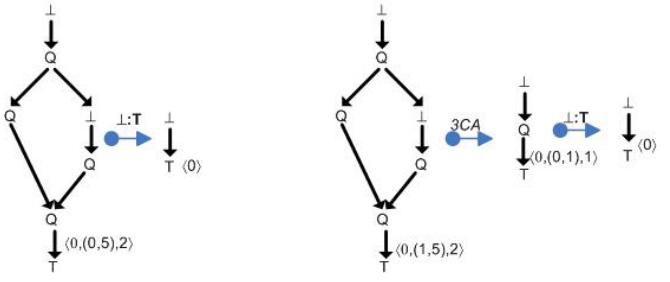


Fig. 9. Normalization proof - different overlapped cut with convergent critical pair

Additionally we emphasize a remarkable difference in the above example. Note at right of Fig. 9 we have been chosen the maximum cut and generated a *valid* reduction strategy. However, at left of Fig. 9 we have not been chosen the maximum cut. Even though we have obtained a reduction strategy where it's normalization measure has been decreased and a cut free N-Graph has been obtained. But the reduction strategy used at left can not be said as a *valid* reduction strategy, (see Def. 11), since it does not start by selecting the maximum cut structure from the proof-graph. As a result, we need to define a slightly different kind of reduction strategy named **quasi-valid reduction strategy**. A quasi-valid reduction strategy in a proof-graph G is determined by not choosing the maximum cut structure C of G. But, choosing an overlapped cut of C, that is C'. Then, we apply the transformation of C' and even though the normalization measure of G is decreased.

Theorem 1 (Normalization). *Every cut* N-Graph G *can be transformed into a cut free* N-Graph G'*, where the sets of premisses and conclusions of* G' *are equivalent to the respective sets of* (G)*.*

Proof. The normalization proof determines the following: *termination*, *weak confluence* and demonstrate that every reduction strategy builded in the normalization procedure is, in fact, a valid or quasi-valid reduction strategy. *Termination* property is guarantee, because every valid or quasi-valid reduction strategy is necessarily a finite and uncyclic process. We remark that every N-Graph cut transformation is applied in a single direction. Therefore, a cyclic normalization process is avoided. A *confluence property* is obtained by means of analyzing the overlapped cuts that yield critical pairs. A weak version of confluence property is obtained, since some overlapped reduction strategies generate divergent critical pairs.

The general mechanism of normalization procedure for N-Graphs is defined to demonstrate that only valid or quasi-valid reduction strategies for G are obtained.

The mechanism starts by choosing the maximum cut structure of G or selecting other cut structure of G in case we have *different overlapped cuts*. Then, we apply a cut transformation or 3CA. As a result, the normalization measure of G is decreased. This procedure continues until we obtain a cut free N-Graph G' for G. We demonstrate that the three kinds of reduction strategies lead to a valid or quasi-valid reduction strategy.

The first kind of reduction strategy (*direct*) can be verified by means of an inspection over cut transformations and 3CA. Considering a cut N-Graph G which is immediately transformed into a cut free N-Graph G' or in a cut N-Graph G'' (*where normalization measure*(G'') *is less than normalization measure*(G)) by means of the following transformations: (*1*). **logical-β or structural-β**: the cut node of G is removed. (*2*). **weakening permutative cuts**: the cut path of G is removed or decreased. (*3*). $\bot$:$\top$ or **extended** $\bot$:$\top$ **cuts**: the cut hole of G is removed. (*4*). 3CA: the cut cycle of G is removed. The second and third kinds of reduction strategies, respectively, *hidden* and *overlapped* strategies have been illustrated in the aforementioned examples.

Observational Completeness on Abstract Interpretation

Gianluca Amato and Francesca Scozzari

Dipartimento di Scienze, Università di Chieti-Pescara
{amato,scozzari}@sci.unich.it

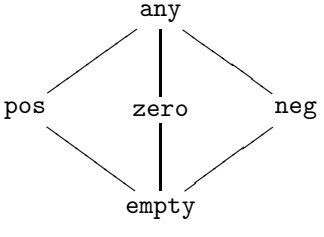
Abstract. In the theory of abstract interpretation, we introduce the observational completeness, which extends the common notion of completeness. A domain is complete when abstract computations are as precise as concrete computations. A domain is observationally complete for an observable π when abstract computations are as precise as concrete computations, if we only look at properties in π. We prove that continuity of state-transition functions ensures the existence of the least observationally complete domain. When state-transition functions are additive, the least observationally complete domain boils down to the complete shell.

1 Introduction

Abstract Interpretation. Abstract interpretation [3,4] is a general theory for approximating the behavior of a discrete dynamic system. The idea is to replace the formal semantics of a system with an abstract semantics, computed over a domain of abstract objects. There are many different methods to describe the semantics of a system. Most of them are based on a partially ordered set (poset) $\langle C, \leq_C \rangle$ of states and a set F of monotone state-transition functions $f : C \to C$. The semantics $\mathcal{S}$ is defined as the (least) fixpoint of a semantic function $\mathcal{F}$ obtained as a composition of state-transition functions. The poset $\langle C, \leq_C \rangle$ is called *concrete domain* and $\mathcal{S} = \operatorname{lfp} \mathcal{F}$ is called the *concrete semantics*.

An abstract interpretation is specified by the poset $\langle A, \leq_A \rangle$ of *abstract objects*. The abstract objects describe the properties of the system we are interested in. The relationship between the concrete and abstract objects is formalized by a monotone concretization map $\gamma : A \to C$ which, given a property $a \in A$, yields the biggest concrete state $c \in C$ which enjoys the property a. Therefore, a property a is a correct approximation of a concrete state c when $c \leq_C \gamma(a)$.

For instance, consider the concrete domain $\wp(\mathbb{Z})$ with the standard ordering given by inclusion, and $Sign = \{\texttt{empty}, \texttt{pos}, \texttt{neg}, \texttt{zero}, \texttt{any}\}$ ordered as depicted on the right. The intuition is that `pos` represents the set of (strictly) positive integers, `zero` represents the singleton $\{0\}$, while `empty` represents the empty set of integers. This may be formalized



H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 99–112, 2009.

by defining γ as follows: $\gamma(\texttt{empty}) = \emptyset$, $\gamma(\texttt{pos}) = \{n \in \mathbb{Z} \mid n > 0\}$, $\gamma(\texttt{neg}) = \{n \in \mathbb{Z} \mid n < 0\}$, $\gamma(\texttt{zero}) = \{0\}$, $\gamma(\texttt{any}) = \mathbb{Z}$.

Often, it is possible to define a monotone abstraction map $\alpha : C \to A$ which yields the largest properties a enjoyed by a concrete object c, such that $c \leq_C \gamma(a) \iff \alpha(c) \leq_A a$. In the previous example, the abstraction is given by $\alpha(c) = \{a \in A \mid c \leq_C \gamma(a)\}$. For instance, $\alpha(\{-1, -2\}) = \texttt{neg}$ while $\alpha(\{-1, 0\}) = \texttt{any}$.

An *abstract domain* is given by the poset A of the abstract objects and the pair of maps $\langle \alpha, \gamma \rangle$. However, since γ is uniquely determined by α and viceversa, in the following we specify an abstract domain just by giving α.

The goal of any abstract interpretation is to compute $\alpha(\mathcal{S})$, that is to find out the properties enjoyed by the semantics of the system. Instead of computing $\mathcal{S}$ (which is not computable) and then applying α, the idea is to replace, in the the definition of $\mathcal{F}$, every state-transition function f with an abstract counterpart $f^\# : A \to A$, which must be *correct*. We say that $f^\#$ is correct if, whenever a is a correct approximation of c, then $f^\#(a)$ is a correct approximation of $f(c)$. This is equivalent to say that any abstract computation $f^\#(\alpha(c))$ approximates the corresponding concrete computation $f(c)$, i.e.:

$$\alpha \circ f \leq_{A \to A} f^\# \circ \alpha \ , \tag{1}$$

where $\leq_{A \to A}$ is the pointwise extension of $\leq_A$. In particular, there is a *best correct abstraction* of f, denoted by f^α, which is $f^\alpha = \alpha \circ f \circ \gamma$. If we replace the state-transition functions in the definition of $\mathcal{F}$ with the corresponding best-correct abstractions, we obtain a new semantic function $\mathcal{F}^\#$ and a new abstract semantics $\mathcal{S}^\# = \operatorname{lfp} \mathcal{F}^\#$, and the theory of abstract interpretation ensures that

$$\alpha(\mathcal{S}) \leq_{\mathcal{A}} \mathcal{S}^\# \ . \tag{2}$$

As an example of abstract state-transition function, consider $inc : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ such that $inc(X) = \{n + 1 \mid n \in X\}$. The best correct abstraction of inc is

$$inc^\alpha(a) = \begin{cases} \texttt{empty} & \text{if } a = \texttt{empty}, \\ \texttt{pos} & \text{if } a = \texttt{zero} \text{ or } a = \texttt{pos}, \\ \texttt{any} & \text{otherwise.} \end{cases}$$

Completeness. Generally speaking, the inequalities (1) and (2) are strict. This means that computing in the abstract domain is (strictly) less precise than computing on the concrete one. For instance, $\alpha(inc(-1)) = \texttt{zero}$ but $inc^\alpha(\alpha(-1)) = \texttt{any}$. When $\alpha \circ f = f^\alpha \circ \alpha$ we say that the abstract domain is *complete* for the function f. Intuitively, when this happens, the best correct abstraction f^α perfectly mimics the concrete function f. For example, given $sq(X) = \{x^2 \mid x \in X\}$, the best correct abstraction is

$$sq^\alpha(a) = \begin{cases} \texttt{pos} & \text{if } a = \texttt{pos} \text{ or } a = \texttt{neg}, \\ a & \text{otherwise.} \end{cases}$$

It follows that $sq(\{-1,-2\}) = \{1,4\}$, and its abstraction is $\alpha(sq(\{-1,-2\})) = \texttt{pos}$, meaning that the square of any integer in $\{-1,-2\}$ is positive. The same result may be obtained by first abstracting $\{-1,-2\}$ and then computing sq^α, since $sq^\alpha(\alpha(\{-1,-2\})) = sq^\alpha(\texttt{neg}) = \texttt{pos}$. It is easy to show that, for any set of integers $X \in \wp(\mathbb{Z})$, it holds that $sq^\alpha(\alpha(X)) = \alpha(sq(X))$. Thus, the abstract domain *Sign* is complete for the function sq.

Completeness enjoys many good properties. If an abstract domain α is complete for f and g, then it holds that:

- α is complete for $f \circ g$ and $f^\alpha \circ g^\alpha = (f \circ g)^\alpha$;
- $\alpha(\mathrm{lfp}\, f) = \mathrm{lfp}(f^\alpha)$.

This implies that (2) is an equality, and therefore one does not lose precision by computing on the abstract domain.

When an abstract domain α is not as precise as the concrete one, that is, the abstract semantics $\mathcal{S}^\#$ does not coincide with $\alpha(\mathcal{S})$, then we need to refine the abstract domain α. This means to replace α by a new domain β and $\mathcal{S}^\#$ by a new abstract semantics $\mathcal{S}^\circ$, such that $\alpha(\mathcal{S})$ may be recovered by $\mathcal{S}^\circ$. Here, $\mathcal{S}^\circ$ is obtained by replacing all the state-transition functions f in $\mathcal{F}$ with f^β. Conceptually, the domain β is the *computational domain* and α is the *observational domain*, which contains all the properties we want to observe. The abstract objects in β which do not belong to α are only used to compute intermediate steps in order not to lose precision. Obviously, we want to keep β as small as possible.

In the literature of abstract interpretation, the standard way of refining α is to compute the least complete domain for F which includes α. This is called the *complete shell* of α, and may be constructively computed [8].

The Goal. In this paper, we show that the complete shell may not be the smallest abstract domain which enables us to recover the property $\alpha(\mathcal{S})$. This is because we are only interested in properties in α, and not in the new objects introduced by β. This observation suggests another notion of completeness, which we call *observational completeness*. A domain β is observationally complete for a function f and an observational domain α when every concrete computation may be approximated in β without losing precision on the properties in α. In order to formalize the observational completeness, we first need to introduce a new ordering between abstract domains. We say that an abstract domain β is *more precise than* an abstract domain β' for observing properties in α whenever the result of each computation on β observed on α is approximated by the result of the corresponding computation on β', observed on α. We show that, under suitable conditions, there exists the smallest observationally complete domain for a given set F of functions and an observational domain. We prove that any complete domain which contains α is also observationally complete for α, but the converse does not hold. We give the conditions under which the least observationally complete domain corresponds to the complete shell.

Plan of the Paper. The next section recalls some basic definitions and notations about abstract interpretation. In Sect. 3 we define the notion of

observational completeness, in Sect. 4 we study the relationships between observational completeness and standard completeness. In Sect. 5 we briefly compare observational completeness to other notions of completeness in the literature, such as forwards completeness and fixpoint completeness.

2 Basic Notions of Abstract Interpretation

In the abstract interpretation theory, abstract domains can be equivalently specified either by Galois connections or by upper closure operators (ucos) [4]. When an abstract domain A is specified by a Galois connection, i.e., a pair of abstraction and concretization maps $\langle \alpha, \gamma \rangle$, then $\gamma \circ \alpha \in uco(C)$ is the corresponding uco on C. On the contrary, given an uco ρ, the corresponding Galois connection is $\langle \rho, id \rangle$. In the rest of the paper, we will use ucos, since they are more concise. Moroever, we assume that the concrete domain C is a complete lattice, which is a standard hypothesis in the abstract interpretation theory.

An uco ρ on the concrete domain C is a monotone, idempotent (i.e., $\rho(\rho(x)) = \rho(x)$) and extensive (i.e., $\rho(x) \geq x$) operator on C. Each uco ρ on C is uniquely determined by the set of its fixpoints, which is its image, i.e. $\rho(C) = \{x \in C \mid \rho(x) = x\}$, since $\rho = \lambda x. \bigwedge \{y \in C \mid y \in \rho(C), x \leq y\}$. Moreover, a subset $X \subseteq C$ is the set of fixpoints of an uco on C iff X is meet-closed, i.e. $X = \mathcal{M}(X) = \{\wedge Y \mid Y \subseteq X\}$. For any $X \subseteq C$, $\mathcal{M}(X)$ is called the Moore-closure of X. Often, we will identify closures with their sets of fixpoints. This does not give rise to ambiguity, since one can distinguish their use as functions or sets according to the context. It is well known that the set $uco(C)$ of all ucos on C, endowed with the pointwise ordering $\supseteq$, gives rise to a complete lattice. The top on $uco(C)$ is $\{\top_C\}$, the bottom is C, and the join operation is set intersection $\cap$. The ordering on $uco(C)$ corresponds to the standard order used to compare abstract domains: A_1 is more concrete than A_2 (or A_2 is more abstract than A_1) iff $A_1 \supseteq A_2$ in $uco(C)$.

An abstract domain $\rho \in uco(C)$ is complete for f iff $\rho \circ f = \rho \circ f \circ \rho$ holds. Giacobazzi et al. [8] give a constructive characterization of complete abstract domains, under the assumption of dealing with continuous concrete functions. A function $f : C \to C$ is (Scott) continuous if it preserves least upper bounds of chains in C, i.e., $f(\bigvee B) = \bigvee f(B)$ for any chain $B \subseteq C$. The idea is to build the greatest (i.e., most abstract) domain in $uco(C)$ which includes a given domain ρ and which is complete for a set $F \subseteq C \to C$ of continuous state-transition functions, i.e., for each function in F. In particular, [8] define a mapping $\mathcal{R}_F : uco(C) \mapsto uco(C)$ as follows:

$$\mathcal{R}_F(\rho) = \mathcal{M}\left(\bigcup_{f \in F, a \in \rho} \max(\{x \in C \mid f(x) \leq a\})\right) \ ,$$

where $\max(X)$ is the set of maximal elements in X. They prove that the most abstract domain which includes ρ and is complete for F is $gfp(\lambda\eta.\mathcal{M}(\rho \cup \mathcal{R}_F(\eta)))$. This domain is called the *complete shell* of ρ for F.

3 Observational Completeness

In abstract interpretation, it is common that, in order to observe a property π with a good deal of precision, we need to perform the computation in a richer domain $\rho \supseteq \pi$. In the following, we call π the *observational domain* and ρ the *computational domain*. In the rest of the paper, we assume given a complete lattice C (the concrete domain), a set F of monotone functions from C to C and an uco $\pi \subseteq C$ which represents the set of observable properties.

A common problem is to find a domain ρ such that if we perform any computation on ρ and we project over π, we obtain the same result of the concrete computation, projected over π. In order to formalize this notion, we first need to define the concept of computation (on both an abstract and a concrete domain).

Definition 1 (Computation). *A finite sequence $\xi = \langle f_1, \ldots, f_n \rangle$ of elements of F is called* computation. *Given a computation ξ, a domain α, and an element $c \in C$, we denote by $\xi^\alpha(c)$ the value $(\alpha \circ f_1 \circ \ldots \circ \alpha \circ f_n)(\alpha(c))$. As a special case, when ξ is the empty computation, we define $\xi^\alpha(c) = \alpha(c)$.*

Note that, if id is the identity abstraction, then $\xi^{id}(c) = (f_1 \circ \ldots \circ f_n)(c)$. We write $\xi(c)$ as a short form for $\xi^{id}(c)$.

We are now able to compare abstract domains in terms of precision of their computations. We say that a domain α is *more precise than* a domain β if it is the case that, the result of a computation on α projected over π is more precise (it is approximated by) the result of the corresponding computation on β projected over π.

Definition 2 (More Precise than). *We say that α is* more precise than *β for computing F observing π, and we write it as $\alpha \leq \beta$, when*

$$\pi\xi^\alpha(c) \leq \pi\xi^\beta(c)$$

for every computation ξ and $c \in C$.

Although the relation $\leq$ depends on F and π, we prefer to use just $\leq$ instead of a more precise notation such as $\leq_F^\pi$, in order to avoid a cumbersome notation. Since F and π are fixed, this does not cause ambiguities.

It is easy to check that $\leq$ is a preorder, which may be viewed as a generalization of the standard ordering between ucos: if $\alpha \supseteq \beta$ then $\alpha \leq \beta$. Our notion is more general than the standard ordering since it allows us to compare two different domains (α and β) w.r.t. their precision on a third domain (π), and does not require neither α nor β to be in any relation with π. Note that, if $\pi = id$, then $\alpha \leq \beta$ iff $\alpha \supseteq \beta$, since we also consider the empty computation.

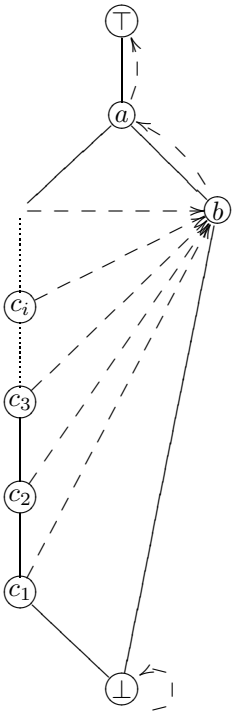
Our formal notion of precision suggests to define a corresponding notion of completeness. We say that a domain α is *observationally complete* for π if any computation on α projected over π, gives the same result of the corresponding concrete computation, projected over π. Here, the key notion is that any computation is always observed on π.

Definition 3 (Observational Completeness). *We say that a domain α is* observationally complete *(for F and π) if α is more precise than the concrete domain, i.e., $\alpha \leq id$.*

Among all the observationally complete domains, we are interested in the least (most abstract) one w.r.t. set inclusion. In general, the least observationally complete domain does not exist, as the following example shows.

Example 1. Let us consider the diagram on the right, where the nodes are the elements of the domain $C = \{\top, \bot, a, b, c_1, c_2, \ldots, c_i, \ldots\}$, solid and dotted edges represent the ordering on C and dashed arrows represent a function $f : C \to C$.

Let $\pi = \{\top, a\}$, $\rho_1 = \{\top, a, b, \bot\} \cup \{c_i \mid i \text{ is even}\}$ and $\rho_2 = \{\top, a, b, \bot\} \cup \{c_i \mid i \text{ is odd}\}$. It is easy to check that both ρ_1 and ρ_2 are observationally complete. However, $\rho = \rho_1 \cap \rho_2 = \{\top, a, b, \bot\}$ is not observationally complete, since, for the computation $\xi = \langle f \rangle$, we have that $\pi(\xi(c_1)) = a$ while $\pi(\xi^{\rho}(c_1)) = \pi(\rho(f(\rho(c_1)))) = \pi(\rho(f(a)) = \top$. □



As a key result we show that, if all the functions in F are continuous, the least observationally complete domain exists.

Theorem 1. *If F is a set of continuous functions, than*

$$\sigma = \mathcal{M}\Big(\bigcup\{\max\{x \in C \mid \xi(x) \leq a\} \mid \xi \text{ computation and } a \in \pi\}\Big)$$

is the least observationally complete domain (for F and π).

In order to exploit this notion of observational completeness for approximating the formal semantics $\mathcal{S}$, we need to show that an observationally complete domain σ preserves the least fixpoint of any composition of functions from F. This result implies that, we can safely approximate the concrete semantic function $\mathcal{F}$ with the abstract semantic function on σ without losing precision on π.

Theorem 2 (Fixpoint Preservation). *Let α be observationally complete for F and π. Then α preserves the least fixpoint of any composition of functions from F, when observing π. In formulas, we have that:*

$$\forall f_1, \ldots f_n \in F,\ \pi(\mathrm{lfp}(f_1 \circ \ldots \circ f_n)) = \pi(\mathrm{lfp}(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)) \ .$$

The previous theorem allows us to say that $\pi(\mathrm{lfp}(\mathcal{F})) = \pi(\mathrm{lfp}(\mathcal{F}^{\#}))$ for any observationally complete domain. In other words, if we only want to observe π, an observationally complete domain does not lose precision in the fixpoint computation involving any composition of functions from F.

4 Observational Completeness and Complete Shell

In this section we study the relationships between observational completeness and the standard notion of completeness, in particular between the least observationally complete domain and the complete shell.

It is immediate to show that if α is complete for F and $\alpha \supseteq \pi$, then α is observationally complete for F and π. More generally, α is observationally complete for F and α. We wonder whether:

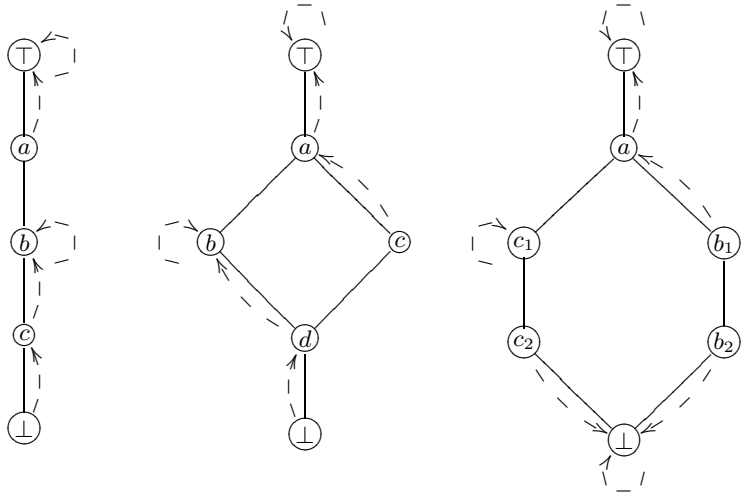
a) every observationally complete domain for F and π is complete for F;
b) the least observationally complete domain for F and π is the complete shell of π for F.

With respect to the first question, note that if α is observationally complete for F and π, every $\beta \supseteq \alpha$ is still observationally complete. This does not hold for completeness: α may be complete for F, although some $\beta \supseteq \alpha$ may not. This is because in the observational completeness the observable properties remain fixed when we refine the initial domain, while for standard completeness the notion of observational domain coincides with the computational domain.

Example 2. Consider the concrete domain $C = \{\top, a, b, c, \bot\}$ depicted in Fig. 1a. The domain $\alpha = \{\top, a, b\}$ is complete for the function depicted in the diagram, hence it is also observationally complete for $\pi = \{\top, a\}$. However, the domain $\{\top, a, b, c\}$ is observationally complete for π but it is not complete. □

Since completeness implies observational completeness, we may argue that the least observationally complete domain for π and F coincides with the complete shell of π. In the general case this is not true and the least observationally complete domain is more abstract than the complete shell. The next examples illustrate this case.

Example 3. Consider the concrete domain $C = \{\top, a, b, c, d, \bot\}$ depicted in Fig. 1b. Assume $\pi = \{\top, a\}$. If we build the complete shell of π for F, in the first step we include the element b and c, since they are the maximal $x \in C$ such that $f(x) \leq a$, and the element d since it is the meet of b and c. At the second step, we also include $\bot$, which is the greatest element $x \in C$ such that $f(x) \leq c$. Note that, in each step, we consider all the elements generated in the previous steps, forgetting the observational domain π which started the process. However, it is trivial to check that the domain $\alpha = C \setminus \{\bot\}$ has the same precision of C when observing π, i.e. $\pi f^i(x) = \pi(\alpha f \alpha)^i(x)$ for every $x \in C$ and $i \in \mathbb{N}$. When $x \in \alpha$, the stronger property $f^i(x) = (\alpha f \alpha)^i(x)$ holds. When $x = \bot$, it is not true that $f^i(\bot) = (\alpha f \alpha)^i(\bot)$: for example it does not hold for $i = 1$. However, for each i, $\pi f^i(\bot) = a = \pi(\alpha f \alpha)^i(\bot)$, hence α is observationally complete. □



(a) Example 2 (b) Example 3 (c) Example 4

Fig. 1. Counterexamples

Example 4. Consider the concrete domain $C = \{\top, a, b_1, b_2, c_1, c_2, \bot\}$ depicted in Fig. 1c. If $\pi = \{\top, a\}$, the complete shell is the entire domain C. However, c_2 is useless when observing π, since the least observationally complete domain is $C \setminus \{c_2\}$. □

4.1 The Case of Additive Functions

We will show that, when all the functions in F are (completely) additive, the least observationally complete domain and the complete shell coincide. However, in order to prove this result, we need to give an alternative construction for the complete shell, more similar to the construction for the least observationally complete domain. In more details, we replace the standard refinement operator $\mathcal{R}_F : uco(C) \mapsto uco(C)$ given in Sect. 2 with a new operator $\hat{\mathcal{R}}_F : \wp(C) \mapsto \wp(C)$ simply obtained by removing the Moore closure from the definition of $\mathcal{R}_F$. Therefore, we define

$$\hat{\mathcal{R}}_F(X) = \bigcup_{f \in F, a \in X} \max\{x \in C \mid f(x) \leq a\} \ .$$

Note that $\hat{\mathcal{R}}_F(X)$ may not be an uco even if X is an uco, and that $\mathcal{R}_F(X) = \mathcal{M}\left(\hat{\mathcal{R}}_F(X)\right)$. We recall that, given a function $G : \wp(C) \to \wp(C)$, we have that $G^\omega(X) = \cup_{i \in \mathbb{N}} G^i(X)$ where $G^0(X) = X$ and $G^{i+1}(X) = G(G^i(X))$.

Theorem 3. *For every set F of continuous maps, the complete shell of π for F is given by*

$$S = \mathcal{M}\left(\hat{\mathcal{R}}_F^\omega(\pi)\right) \ .$$

This new construction, which is the key result to prove Theorem 4, is interesting in itself, since it sheds a new light on the construction of the complete shell. First of all, it shows that it is not necessary to compute the Moore closure at each step of the refinement, but it suffices to compute it at the end. Secondly, it shows that we need at most ω steps of refinement to reach the fixpoint.

We recall that a function $f : C \to C$ is (completely) additive if it preserves arbitrary least upper bounds, i.e., $f(\bigvee B) = \bigvee f(B)$ for any $B \subseteq C$.

Theorem 4. *If F is a set of completely additive functions, the complete shell S of π for F is the smallest observationally complete domain σ for F and π.*

It is worth noting that, even if F is a set of additive functions, this theorem does not imply that observational completeness and completeness are the same thing: in Example 2 the function f is additive, yet there is an observationally complete domain which is is not complete.

5 Conclusions and Related Work

Different kinds of completeness have been proposed in the literature. The first notion of completeness appears in Cousot and Cousot [4]. In the same paper, the notion of *fixpoint completeness* is formalized. A domain α is fixpoint complete for a function f when it preserves the least fixpoint of f, in formulas $\alpha(\mathrm{lfp}\, f) = \mathrm{lfp}(\alpha \circ f \circ \alpha)$. Cousot and Cousot have shown that complete domains are also fixpoint complete. A detailed study on completeness and fixpoint completeness can be found in Giacobazzi et al. [8], where the authors solve the problem of synthesizing complete abstract domains.

Cousot and Cousot [2] introduced a different notion of completeness called *exactness*. The same notion has been renamed as *forward completeness* (F-completeness) by Giacobazzi and Quintarelli [7] who apply the completeness results on model checking. Moreover, to distinguish between standard completeness and F-completeness, Giacobazzi and Quintarelli renamed the former as *backward completeness* (B-completeness). A domain α is F-complete for a function f when $f \circ \alpha = \alpha \circ f \circ \alpha$. Intuitively, this means that the result of any abstract computation coincides with the result of the corresponding concrete one, when the starting object is an abstract object.

Our notion of observational completeness differs from all the previous notions (B-completeness, fixpoint completeness, F-completeness). The main point is that, in our notion, we have two concepts of observational and computational domain and, most importantly, the observational domain is kept fixed when refining. We believe that, in any static analysis or semantics definition, the observable property does not change when looking for better domains. On the contrary, B-completeness is self-referential, since the observational domain changes when refining the domain. More precisely, given a domain π, the complete shell of π for f is the least abstract domain β containing π which is observationally complete for β (and thus it is observationally complete for π). Moreover, the self-referentiality of completeness yields some counter-intuitive

behaviors. For instance, if α is complete and β contains α, it may well happen that β is not complete even if, according to our intuition, β is "richer" than α. This does not happen for observational completeness, where supersets of observationally complete domains are still observationally complete (see Example 2).

The notion of F-completeness does not fix any observable property. This kind of completeness is useful when we are interested in a subset of the concrete domain closed for the application of any state-transition function.

Finally, fixpoint completeness does not take into consideration intermediates steps during the abstract computation. In fact, it is only required that the abstract least fixpoint (computed on the abstract domain) coincides with the abstraction of the concrete least fixpoint. Giacobazzi et al. [8] show that, even under strong hypotheses, the existence of the least fixpoint complete domain containing π cannot be ensured. They show that, even if the concrete domain is a complete Boolean algebra or a finite chain, and the concrete function f is both additive and co-additive, the least fixpoint complete domain containing π does not necessarily exist. The counterexamples suggest that finding reasonable conditions for the existence of least fixpoint complete domains is not viable.

Other notions of completeness have been proposed for dealing with logic (see, for instance, Cousot and Cousot [5], Schmidt [9], Dams et al. [6]). In general, completeness problems on fragments of temporal logic are considered (covering, preservation, strong preservation). All these notions are very different from the other forms of completeness, since they consider only fixed logical operators, and, in general, one is not interested in least fixpoint preservation.

As a future work, we think that observational completeness could be generalized, in order to be relative to an abstract domain, instead of the concrete one. We say that a domain α is observationally complete for π and F relatively to the domain β, when the result of any abstract computation, observed over π, is more precise than the corresponding abstract computation on the domain β, observed over π. Here, the novelty is that the domains α, β and π do not need to be in any relation. Thus, the least observationally complete domain for π and F relatively to β could be incomparable with β.

Observational completeness naturally arises once we fix the preorder $\leq$ on domains, which formalizes the intuitive notion of precision. The novelty with respect to the standard treatment of completeness is that we have two orderings on ucos: standard inclusion $\supseteq$ and precision $\leq$. The latter is used to define what an observationally complete domain is, while the former selects, among observationally complete domains, the preferred one. In the complete shell construction the two orderings coincide. In principle, we could change the standard inclusion ordering, obtaining a different notion of "least" observationally complete domain. For instance, we could compare two abstract domains on the base of their cardinality or of a suitable notion of "complexity" of their abstract objects.

References

1. Birkhoff, G.: Lattice Theory, 3rd edn., vol. XXV. AMS Colloquium Publications, American Mathematical Society (1967)
2. Cousot, P.: Types as abstract interpretations, invited paper. In: POPL 1997: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 316–331. ACM Press, New York (1997)
3. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL 1977: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 238–252. ACM Press, New York (1977)
4. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: POPL 1979: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, pp. 269–282. ACM Press, New York (1979)
5. Cousot, P., Cousot, R.: Temporal abstract interpretation. In: POPL 2000: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 12–25. ACM Press, New York (2000)
6. Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems. ACM Transactions on Programming Languages and Systems 19(2), 253–291 (1997)
7. Giacobazzi, R., Quintarelli, E.: Incompleteness, counterexamples, and refinements in abstract model-checking. In: Cousot, P. (ed.) SAS 2001. LNCS, vol. 2126, pp. 356–373. Springer, Heidelberg (2001)
8. Giacobazzi, R., Ranzato, F., Scozzari, F.: Making abstract interpretations complete. Journal of the ACM 47(2), 361–416 (2000)
9. Schmidt, D.A.: Comparing completeness properties of static analyses and their logics. In: Kobayashi, N. (ed.) APLAS 2006. LNCS, vol. 4279, pp. 183–199. Springer, Heidelberg (2006)

A Appendix

In some proofs we will make use of the Hausdorff's maximality principle [1]. We recall that a chain Y in a poset P is maximal (with respect to set inclusion) whenever for any other chain Y' in P, $Y \subseteq Y'$ implies $Y = Y'$. The Hausdorff's maximal principle says that every chain in a poset P can be extended to a maximal chain in P.

Theorem 1. *If F is a set of continuous functions, than*

$$\sigma = \mathcal{M}\Big(\bigcup\{\max\{x \in C \mid \xi(x) \leq a\} \mid \xi \textit{ computation and } a \in \pi\}\Big)$$

is the least observationally complete domain (for F and π).

Proof. First of all, we show that σ is observationally complete. We prove, by induction on the length of ξ, that $\xi(x) \leq a$ implies $\xi^\sigma(x) \leq a$ for each computation ξ and $a \in \pi$. If $|\xi| = 0$, then $\xi(x) = x$ and $\xi^\sigma(x) = \sigma(x)$. Note that $\sigma \supseteq \pi$ since if $a \in \pi$ then $a = \bigvee\{x \in C \mid x \leq a\}$. Hence $x \leq a$ implies $\sigma(x) \leq a$. Now assume $|\xi| = i + 1$. If $\xi(x) \leq a$, consider the poset $C' = \{c \in C \mid \xi(c) \leq a\}$ and the chain $\{x\} \subseteq C'$. By Hausdorff's maximality principle there exists a maximal chain

$Y \supseteq \{x\}$ which is contained in C'. Let $y = \bigvee Y$. By continuity of ξ, we have that $\xi(y) \leq a$. Since Y is a maximal chain in C', then $y \in \max C'$. Moreover, by definition of σ we have that $y \in \sigma$. It follows that $\xi(\sigma(x)) \leq \xi(y) \leq a$. If $\xi = \xi_1 \cdot f$, by inductive hypothesis $\xi^\sigma(x) = \xi_1^\sigma(f(\sigma(x))) \leq a$ since $\xi_1(f(\sigma(x))) \leq a$.

Now we show that σ is the least observationally complete domain. Assume, without loss of generality, that $v \in \max\{x \in C \mid \xi(x) \leq a\}$ for some computation ξ and $a \in \pi$ and let ρ be a domain such that $v \notin \rho$. Then, $\xi(\rho(v)) \not\leq a$ since $\rho(v) > v$ and by definition of v. Hence, also $\xi^\rho(v) \not\leq a$, which means ρ is not observationally complete. □

Theorem 2 (Fixpoint Preservation). *Let α be observationally complete for F and π. Then α preserves the least fixpoint of any composition of functions from F, when observing on π. In formulas, we have that:*

$$\forall f_1, \ldots f_n \in F,\ \pi(\mathrm{lfp}(f_1 \circ \ldots \circ f_n)) = \pi(\mathrm{lfp}(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)) \ .$$

Proof. It clearly holds that

$$\pi(\mathrm{lfp}(f_1 \circ \ldots \circ f_n)) \leq \pi(\mathrm{lfp}(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)) \ ,$$

since α is extensive. We now show the other direction. We prove that, for any $c \in C$ and ordinal ϵ, it holds that

$$\pi\left(\bigvee_{i\in\epsilon}(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)^i(c)\right) \leq \pi\left(\bigvee_{i\in\epsilon}(f_1 \circ f_2 \circ \ldots \circ f_n)^i(c)\right) \ .$$

Since π is an upper closure operator, it is complete for arbitrary lubs. It follows that:

$$\pi\left(\bigvee_{i\in\epsilon}(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)^i(c)\right) = \pi\left(\bigvee_{i\in\epsilon}\pi(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)^i(c)\right) \ .$$

Since α is observationally complete for F and π, then

$$\pi\left(\bigvee_{i\in\epsilon}\pi(\alpha \circ f_1 \circ \alpha \circ f_2 \circ \ldots \alpha \circ f_n \circ \alpha)^i(c)\right) = \pi\left(\bigvee_{i\in\epsilon}\pi(f_1 \circ f_2 \circ \ldots \circ f_n)^i(c)\right) \ ,$$

which is equivalent to $\pi\left(\bigvee_{i\in\epsilon}(f_1 \circ f_2 \circ \ldots \circ f_n)^i(c)\right)$. □

Lemma 1. *For every set F of continuous functions and every $X \subseteq C$, we have*

$$\mathcal{M}\left(\hat{\mathcal{R}}_F(\mathcal{M}(X))\right) = \mathcal{M}\left(\hat{\mathcal{R}}_F(X)\right) \ .$$

Proof. It is immediate by monotonicity of $\hat{\mathcal{R}}_F$ and $\mathcal{M}(\_)$ that $\mathcal{M}\left(\hat{\mathcal{R}}_F(\mathcal{M}(X))\right) \supseteq \mathcal{M}\left(\hat{\mathcal{R}}_F(X)\right)$. For the converse inequality, since $\mathcal{M}(\_)$ is an upper closure operator on $\wp(C)$, it is enough to prove that $\hat{\mathcal{R}}_F(\mathcal{M}(X)) \subseteq \mathcal{M}\left(\hat{\mathcal{R}}_F(X)\right)$. Given

$y \in \hat{\mathcal{R}}_F(\mathcal{M}(X))$, we have $y \in \max\{x \in C \mid f(x) \leq a\}$ and $a = \bigwedge_{i\in I} a_i$ where $f \in F$ and $\{a_i\}_{i\in I} \subseteq X$.

For each $i \in I$, consider the set $Y_i = \max\{x \in C \mid f(x) \leq a_i\} \subseteq \hat{\mathcal{R}}_F(X)$. Since $f(y) \leq a \leq a_i$ and f is continuous, there is an $y_i \in Y_i$ such that $y_i \geq y$. We may find y_i as the least upper bound of a maximal chain in Y_i containing y, which exists by Hausdorff's maximality principle. It is enough to prove that $y = \bigwedge_{i\in I} y_i$. By definition of the y_i's, we have $y \leq \bigwedge_{i\in I} y_i$. Moreover, $f(\bigwedge_{i\in I} y_i) \leq f(y_i) \leq a_i$ hence $f(\bigwedge_{i\in I} y_i) \leq a$. Since y is a maximal element such that $f(y) \leq a$, this means that $y = \bigwedge_{i\in I} y_i$. □

Theorem 3. *For every set F of continuous maps, the complete shell S of π for F is given by*

$$S = \mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) .$$

Proof. It can be easily proved that

$$S = \mathcal{M}\left(G^{\kappa}(\{\top\})\right)$$

for some ordinal κ, where $G : uco(C) \mapsto uco(C)$ is the map

$$G(\rho) = \mathcal{M}\left(\pi \cup \mathcal{R}_F(\rho)\right) = \mathcal{M}\left(\pi \cup \hat{\mathcal{R}}_F(\rho)\right) .$$

It is enough to prove that $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)$ is a subset of S and a fixpoint of G. In order to prove $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) \subseteq S$ it is enough to show that $\mathcal{R}_F^i(\pi) \subseteq G^{i+1}(\{\top\})$ for every $i < \omega$. The proof is by induction over i. For $i = 0$, we have $\hat{\mathcal{R}}_F^0(\pi) = \pi \subseteq G(\{\top\})$. If $i = j + 1$, $\hat{\mathcal{R}}_F^i(\pi) = \hat{\mathcal{R}}_F(\hat{\mathcal{R}}_F^j(\pi)) \subseteq \hat{\mathcal{R}}_F(G^i(\{\top\})) \subseteq G(G^i(\{\top\})) = G^{i+1}(\{\top\})$. Now we prove that $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)$ is a fixpoint of G. We have that

$$\begin{aligned} G\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right) &= \mathcal{M}\left(\pi \cup \hat{\mathcal{R}}_F\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right)\right) \\ &= \mathcal{M}\left(\pi \cup \mathcal{M}\left(\hat{\mathcal{R}}_F\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right)\right)\right) \\ &= \mathcal{M}\left(\pi \cup \mathcal{M}\left(\hat{\mathcal{R}}_F(\hat{\mathcal{R}}_F^{\omega}(\pi))\right)\right) \\ &= \mathcal{M}\left(\pi \cup \bigcup\{\hat{\mathcal{R}}_F^{i+1}(\pi) \mid i < \omega\}\right) \\ &= \mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) . \end{aligned}$$

This concludes the proof. □

Theorem 4. *If F is a set of completely additive functions, the complete shell S of π for F is the least observationally complete domain σ for F and π.*

Proof. We know that S is observationally complete, since it is complete and contains π. It is enough to prove that if $a \in \hat{\mathcal{R}}_F^{\omega}(\pi)$ and $a \notin \rho$, then ρ is not observationally complete for π and F. Assume $a \in \hat{\mathcal{R}}_F^{i}(\pi)$ and there is no $j < i$ such that $a \in \hat{\mathcal{R}}_F^{i}(\pi)$. It means there exist a computation $\xi = \langle f_1, \ldots, f_i \rangle$ of maps in F and a sequence $a_0, \ldots, a_i$ of objects in C such that $a = a_i$, $a_0 \in \pi$ and $a_j \in \max\{x \in C \mid f_j(x) \leq a_{j-1}\}$ for any $j \in [1, \ldots, i]$. It immediate to check that $\xi(a) \leq a_0$. We prove that $\xi^{\rho}(a) \not\leq a_0$.

Note that, if f is completely additive, then $\max\{x \in C \mid f(x) \leq y\}$ is a singleton for any $y \in C$. Therefore, if $\max\{x \in C \mid f(x) \leq y\} = \{z\}$ and $x \not\leq z$, then $f(x) \not\leq y$. In our proof, this means that, for each $j \in [1, \ldots, i]$, $x \not\leq a_j$ implies $f_j(x) \not\leq a_{j-1}$. Since $\rho(f(x)) \geq f(x)$, this also implies $\rho(f_j(a_j)) \not\leq a_{j-1}$. Since $a \notin \rho$, then $\rho(a) > a$, i.e. $\rho(a) \not\leq a$, hence $\xi^{\rho}(a) \not\leq a_0$. □

SAT in Monadic Gödel Logics: A Borderline between Decidability and Undecidability

Matthias Baaz, Agata Ciabattoni*, and Norbert Preining**

Technische Universität Wien, Austria

Abstract. We investigate satisfiability in the monadic fragment of first-order Gödel logics. These are a family of finite- and infinite-valued logics where the sets of truth values V are closed subsets of $[0,1]$ containing 0 and 1. We identify conditions on the topological type of V that determine the decidability or undecidability of their satisfiability problem.

1 Introduction

Monadic logic is a first-order logic in which all predicates are unary. Classical monadic logic is a rather simple fragment: decidable (both the validity and satisfiability problem) and having finite model property. The same does not hold for many-valued monadic logics in which a rather complex landscape appears and many questions are still open, see e.g. [10,9,3].

The family of (finite- and infinite-valued) *Gödel logics* is a prominent example of many-valued logics. Gödel logics, defined in general over sets of truth values V which are closed subsets of $[0,1]$ containing both 0 and 1, are the only many-valued logics which are completely specified by the *order structure* of V [4]. This fact characterizes Gödel logics as logics of comparative truth and renders them an important case of so-called *fuzzy logics*, see [8].

Different choices of V in general induce different Gödel logics, see [13,4,6]. For $V = \{0,1\}$ the resulting logic coincides with classical logic.

Validity for monadic Gödel logic with $V = [0,1]$ was shown to be undecidable in [2]. Note that in contrast with classical logic, in Gödel logics validity and satisfiability are not dual concepts. A general investigation of the (un)decidability status for the validity problem in monadic Gödel logics was carried out in [3], where it was shown that with the possible exception of Gödel logic $\mathbf{G}_\uparrow$ in which $V = \{1 - 1/n : n \geq 1\} \cup \{1\}$, validity is undecidable when V is infinite, even when restricted to prenex formulas.

In this paper we investigate the (1-)satisfiability problem SAT^m in monadic Gödel logics. We identify conditions on the topological type of V which characterize Gödel logics with decidable and with undecidable SAT^m. SAT^m is shown to be decidable for Gödel logics in which 0 is an isolated point in V (i.e., 0 has Cantor-Bendixon rank $|0|_{\mathrm{CB}} = 0$, see e.g. [12]). Finite-valued Gödel logics being

* Supported by Vienna Science and Technology Fund (WWTF), project MA07-016.
** Supported by Austrian science foundation (FWF), project P19872-N13.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 113–123, 2009.

prominent examples (and any witnessed Gödel logic [11] as well as any prenex Gödel logic can be treated in the same way). In the remaining Gödel logics the presence of at least three predicate symbols, one of which is a constant different from 0 or 1, makes SAT^m undecidable. A complex argument is used to show that without this constant predicate, SAT^m remains undecidable for all Gödel logics in which 0 is a limit point of limit points in V (i.e., $|0|_{\mathrm{CB}} \geq 2$). Gödel logic with $V = [0,1]$ being a prominent example. The (un)decidability status of SAT^m is left open for Gödel logics in which $|0|_{\mathrm{CB}} = 1$ in V and no constant predicate is available; this case is shown to contain only one logic: Gödel logic $\mathbf{G}_\downarrow$ in which $V = \{1/n : n \in \mathbb{N}\} \cup \{0\}$. SAT^m of monadic Gödel logics extended with the projection operator $\triangle$ of [1] is also investigated. This operator allows one to recover classical reasoning inside Gödel logics. We show that the addition of $\triangle$ does not affect the decidability of SAT^m in the finite-valued case, while it does in *all* infinite-valued Gödel logics (even in the witnessed case) for which SAT^m is undecidable in presence of $\triangle$.

2 Syntax and Semantics of Gödel Logics

The language of first-order Gödel logics is identical to that of classical logic.

We call any closed set $V \subseteq [0,1]$ which contains 0 and 1 a *Gödel set*. Let V be a Gödel set, we denote with $\mathbf{G}_V$ the Gödel logic based on the set of truth values V. An *interpretation* (or evaluation) $\varphi_{\mathbf{G}_V}$ for $\mathbf{G}_V$ maps constants and object variables to elements of a *domain* D, n-ary function symbols to functions from D^n into D, and n-ary predicate symbol P to a function from D^n into V. (We will use $\varphi_{\mathbf{G}}$ when the truth value set V is clear from the context)

The interpretation $\varphi_{\mathbf{G}_V}$ extends in the usual way to a function mapping all terms of the language to an element of the domain. $\varphi_{\mathbf{G}_V}$ evaluates atomic formulas $Q \equiv P(t_1, \ldots, t_n)$ and the truth constant $\bot$ as

$$\varphi_{\mathbf{G}_V}(Q) = \varphi_{\mathbf{G}_V}(P)(\varphi_{\mathbf{G}_V}(t_1), \ldots, \varphi_{\mathbf{G}_V}(t_n)) \qquad \varphi_{\mathbf{G}_V}(\bot) = 0$$

Extension to all formulas is given by

$$\varphi_{\mathbf{G}_V}(A \to B) = \begin{cases} 1 & \text{if } \varphi_{\mathbf{G}_V}(A) \leq \varphi_{\mathbf{G}_V}(B) \\ \varphi_{\mathbf{G}_V}(B) & \text{otherwise,} \end{cases}$$

$$\varphi_{\mathbf{G}_V}(A \wedge B) = \min(\varphi_{\mathbf{G}_V}(A), \varphi_{\mathbf{G}_V}(B))$$

$$\varphi_{\mathbf{G}_V}(A \vee B) = \max(\varphi_{\mathbf{G}_V}(A), \varphi_{\mathbf{G}_V}(B))$$

$\neg A$, $A \leftrightarrow B$ and $A \prec B$ abbreviate $A \to \bot$, $(A \to B) \wedge (B \to A)$ and $((B \to A) \to B)$, respectively; in particular $\varphi_{\mathbf{G}_V}(A \prec B)$ is 1 iff either $\varphi_{\mathbf{G}_V}(A) < \varphi_{\mathbf{G}_V}(B)$, or $\varphi_{\mathbf{G}_V}(A) = \varphi_{\mathbf{G}_V}(B) = 1$. Hence the formula $A \prec B$ expresses 'strictly less' but at 1 (note that $\varphi_{\mathbf{G}_V}(1 \prec 1) = 1$). Let us define the *distribution* of a formula A and a free variable x with respect to an interpretation $\varphi_{\mathbf{G}_V}$ as $\mathrm{Distr}(A(x)) = \{\varphi'_{\mathbf{G}_V}(A(x)) : \varphi'_{\mathbf{G}_V} \sim_x \varphi_{\mathbf{G}_V}\}$, where $\varphi'_{\mathbf{G}_V} \sim_x \varphi_{\mathbf{G}_V}$ means that $\varphi_{\mathbf{G}_V}$ is exactly as $\varphi_{\mathbf{G}_V}$ with the possible exception of the domain element

assigned to x. The semantics of quantifiers is given by the infimum and supremum of the corresponding distribution, that is

$$\varphi_{\mathbf{G}_V}(\forall x A(x)) = \inf \mathrm{Distr}(A(x)) \qquad \varphi_{\mathbf{G}_V}(\exists x A(x)) = \sup \mathrm{Distr}(A(x)).$$

In the case that all infima and suprema are actually realized by an element, i.e., all infima are minima and all suprema are maxima, we speak about *witnessed Gödel logics*, see e.g. [11].

Definition 1. *A formula A is* valid *or is a* tautology *(resp. a* positive tautology*) in $\mathbf{G}_V$ if for every interpretation $\varphi_{\mathbf{G}_V}$, $\varphi_{\mathbf{G}_V}(A) = 1$ (resp. $\varphi_{\mathbf{G}_V}(A) > 0$). A is* 1-satisfiable*, or simply* satisfiable*, if there is an interpretation $\varphi_{\mathbf{G}_V}$ such that $\varphi_{\mathbf{G}_V}(A) = 1$; otherwise A is said to be* unsatisfiable*.*

In contrast with classical logic, in Gödel logics validity and satisfiability are not dual concepts. However the following relation holds: A is a positive tautology if and only if $\neg A$ is unsatisfiable.

Consider the unary operator $\triangle$ with the following meaning [1]:

$$\varphi_{\mathbf{G}_V}(\triangle A) = \begin{cases} 1 & \text{if } \varphi_{\mathbf{G}_V}(A) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

We will investigate *monadic* Gödel logics (with and without the operator $\triangle$) i.e., in which all predicate symbols are (at most) unary and no function symbol occur. Every Gödel set V induces a logic $\mathbf{G}_V$ over the language without $\triangle$ and a logic $\mathbf{G}_V^{\triangle}$ if $\triangle$ is present. *Standard Gödel logic* is $\mathbf{G}_{[0,1]}$; i.e., the logic over the full real unit interval as truth value set, see, e.g., [8,16]. We use $\mathbf{G}_n$ to denote the n-valued Gödel logic for $n \geq 2$. Let G be a Gödel logic, we denote with TAUT_G^m and SAT_G^m the validity and satisfiability problems of monadic G, respectively.

2.1 Cantor-Bendixon Ranks and Gödel Sets

In Gödel logics, the validity of a formula depends only on the *relative ordering* and the *topological type* of the truth values of atomic formulas, and not on their specific values. It is therefore important to investigate the topological structure of the underlying truth values sets of Gödel logics, see [4,13]. We recall some definitions for the theory of polish spaces, details may be found in [12].

Definition 2. *A* limit point *of a topological space is a point that is not isolated, i.e., for every open neighborhood U of x there is a point $y \subset U$ with $y \neq x$. For any topological space X let $X' = \{x \in X : x \text{ is limit point of } X\}$. We call X' the* Cantor-Bendixon derivative *of X and the operation itself* Cantor-Bendixon derivation*. Using transfinite recursion we define the* iterated Cantor-Bendixon derivatives *X^α, α ordinal, as follows:*

$$X^0 = X \qquad X^{\alpha+1} = (X^\alpha)' \qquad X^\lambda = \bigcap_{\alpha<\lambda} X^\alpha, \lambda \text{ limit ordinal.}$$

For any polish space X, the least ordinal α_0 such that $X^{\alpha_0} = X^\alpha$ for all $\alpha > \alpha_0$ is called the Cantor-Bendixon rank *of X. For any $x \in X$, we define its (Cantor-Bendixon-)rank $|x|_{\mathrm{CB}} = \sup\{\alpha : x \in X^\alpha\}$.*

3 Decidability Results

As is well known, the satisfiability problem for monadic classical logic (i.e., $\mathbf{G}_2$) is decidable, whereas already a single binary predicate symbol leads to undecidability. We show below that this result can be generalized to monadic Gödel logics $\mathbf{G}_V$ in which 0 is an isolated point in V (i.e., $|0|_{\mathrm{CB}} = 0$); satisfiability in these logics, which includes $\mathbf{G}_n$, coincides with satisfiability in monadic $\mathbf{G}_2$. The same result holds also for any witnessed or prenex monadic Gödel logic. The addition of $\triangle$ to $\mathbf{G}_n$ does not change the decidability status of $\mathrm{SAT}^m_{\mathbf{G}_n^{\triangle}}$, though satisfiability is not anymore equivalent to satisfiability in $\mathbf{G}_2$.

Theorem 1. *Let V be any Gödel set with $|0|_{\mathrm{CB}} = 0$. $\mathrm{SAT}^m_{\mathbf{G}_V}$ is equivalent to $\mathrm{SAT}^m_{\mathbf{G}_2}$.*

Proof. Let Q be any formula of $\mathbf{G}_V$. If Q is satisfiable in monadic classical logic then Q is satisfiable in $\mathbf{G}_V$. For the converse direction, consider any interpretation $\varphi_{\mathbf{G}}$ of $\mathbf{G}_V$ such that $\varphi_{\mathbf{G}}(Q) = 1$. An interpretation $\varphi_{\mathbf{CL}}$ of classical logic such that $\varphi_{\mathbf{CL}}(Q) = 1$ is defined as follows: for any atomic formula A

$$\varphi_{\mathbf{CL}}(A) = \begin{cases} 1 & \text{if } \varphi_{\mathbf{G}}(A) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that for each formula P, $(*)$ $\varphi_{\mathbf{G}}(P) = 0$ if and only if $\varphi_{\mathbf{CL}}(P) = 0$ and $\varphi_{\mathbf{G}}(P) > 0$ if and only if $\varphi_{\mathbf{CL}}(P) = 1$. The proof proceeds by induction on the complexity of P and all cases go though for *all* Gödel logics except when P has the form $\forall x P_1(x)$; in this case, being 0 an isolated point in V, $\varphi_{\mathbf{G}}(P) = 0$ if and only if there is an element u in the domain of $\varphi_{\mathbf{G}}$ such that $\varphi_{\mathbf{G}}(P_1(u)) = 0$; by induction hypothesis $\varphi_{\mathbf{CL}}(P_1(u)) = 0$ and hence $\varphi_{\mathbf{CL}}(\forall x P_1(x)) = 0$. □

Corollary 1. *Satisfiability in the existential fragment of all monadic Gödel logics coincides with satisfiability in monadic classical logic.*

Corollary 2. *Satisfiability in monadic witnessed Gödel logics coincides with satisfiability in monadic classical logic.*

Corollary 3. *Satisfiability in prenex*[1] *monadic Gödel logics coincides with satisfiability in monadic classical logic.*

Proof. Let $Q = \mathcal{Q}\bar{x}P$ be any prenex formula, were $\mathcal{Q}\bar{x}$ is the formula prefix and P does not contain quantifiers. The proof is similar to that of Theorem 1. Assume that $\varphi_{\mathbf{G}}(Q) = 1$. As above we can prove $(*)$ for P. $\varphi_{\mathbf{CL}}(Q) = 1$ easily follows by induction on the number n of quantifiers in $\mathcal{Q}\bar{x}$. □

This result contrasts with the undecidability of the validity problem proved in [3] for all prenex monadic Gödel logics with the exception of $\mathbf{G}_n$ and (possibly) $\mathbf{G}_{\uparrow}$.

Corollary 4. $\mathrm{SAT}^m_{\mathbf{G}_n}$ *coincides with* $\mathrm{SAT}^m_{\mathbf{G}_2}$.

[1] In general Gödel logics do not admit equivalent prenex formulas, see e.g. [4].

Consider $\mathbf{G}_n$ extended with $\triangle$; the formula $\neg\neg A \wedge \neg\triangle A$ is satisfiable in $\mathbf{G}_n^{\triangle}$, with $n > 2$ (take any interpretation $\varphi_{\mathbf{G}}$ such that $0 < \varphi_{\mathbf{G}}(A) < 1$) but it is not in classical logic. This shows that satisfiability in $\mathbf{G}_n^{\triangle}$ is not the same as satisfiability in classical logic. Nevertheless we have

Theorem 2. $\mathrm{SAT}^m_{\mathbf{G}_n^{\triangle}}$ *is decidable.*

Proof. Proceed similarly to the decidability proof of $\mathrm{TAUT}^m_{\mathbf{G}_n^{\triangle}}$ in [3]. Given any monadic formula P, it is indeed enough to consider interpretations with domain $\{1, \ldots, n^k\}$, where k is the number of different predicate symbols occurring in P. The number of such interpretations is finite (at most $n^{k \cdot n^k}$). □

4 Undecidability Results

In this section we prove that $\mathrm{SAT}^m_{\mathbf{G}_V}$ is undecidable in the following cases: all infinite-valued Gödel logics with $\triangle$ (Section 4.1), all Gödel logics where 0 is not isolated and there are three predicates, one of which is a constant evaluated between 0 and 1 (Section 4.2), and all Gödel logics where 0 has at least Cantor-Bendixon rank 2, i.e., 0 is a limit point of limit points (Section 4.3).

Our results adapt and generalize the undecidability proof of $\mathrm{TAUT}^m_{\mathbf{G}_{[0,1]}}$ sketched in [7]. Consider a generic formula A in the classical theory **CE** of two equivalence relations $\equiv_1$ and $\equiv_2$. Without loss of generality we can assume that A is in prenex and disjunctive normal form, i.e., A is of the form:

$$A = \mathcal{Q}^* \bigvee_j \bigwedge_k (x_j^k \equiv_i y_j^k)^l$$

where $\mathcal{Q}^*$ is the formula prefix and l is either -1, in which case the atomic equivalence is negated, or 1, in which case it is positive (not negated).

In the subsections below we will translate A into formulas of $\mathbf{G}_V$ by replacing each term $(x_j^k \equiv_i y_j^k)^l$ in A by suitable $\sigma((u_j^k \equiv_i v_j^k)^l)$. As notational extension we write $\sigma(A) = \mathcal{Q}^* \bigvee_j \bigwedge_k \sigma((x_j^k \equiv_i y_j^k)^l)$.

Lemma 1. *Given two interpretations* $\varphi_{\mathbf{CE}}$ *and* $\varphi_{\mathbf{G}}$*. If for each term* u_j^k *and* v_j^k

$$\varphi_{\mathbf{CE}}((u_j^k \equiv_i v_j^k)^l) = 1 \quad \Leftrightarrow \quad \varphi_{\mathbf{G}}(\sigma(u_j^k \equiv_i v_j^k)^l) = 1$$

(with $l = -1, 1$*) and* $\varphi_{\mathbf{G}}(\sigma(u_j^k \equiv_i v_j^k)^l) = 1$ *or* $\varphi_{\mathbf{G}}(\sigma(u_j^k \equiv_i v_j^k)^l) \leq k < 1$ *then*

$$\varphi_{\mathbf{CE}}(A) = 1 \quad \Leftrightarrow \quad \varphi_{\mathbf{G}}(\sigma(A)) = 1.$$

4.1 Infinite-Valued Gödel Logics with $\triangle$

We show the undecidability of $\mathrm{SAT}_{\mathbf{G}_V^{\triangle}}$, where V is any infinite Gödel set. Our argument, which applies also to all infinite-valued witnessed Gödel logics with $\triangle$, is similar to that for the undecidability of $\mathrm{TAUT}_{\mathbf{G}_V^{\triangle}}$ in [3].

Theorem 3. *Let V be any infinite Gödel set. Satisfiability of monadic formulas in $\mathbf{G}^{\triangle}_V$, with at least two predicate symbols, is undecidable.*

Proof. Validity of formulas in the classical theory **CE** of two equivalence relations was shown in [15] to be r.e. but not recursive. As a formula in classical logic is valid if and only if its negation is unsatisfiable, the satisfiability problem in **CE** is undecidable and not even recursively enumerable. The theorem's claim follows by faithfully interpret **CE** in the monadic fragment of $\mathbf{G}^{\triangle}_V$. Indeed, let P_1 and P_2 be different unary predicate symbols in $\mathbf{G}^{\triangle}_V$. A translation $\sigma(A)$ of **CE**-formulas A into monadic $\mathbf{G}^{\triangle}_V$-formulas is simply defined by translating each atomic formula in A as follows: $\sigma(x^k_j \equiv_i y^k_j) = \triangle(P_i x \leftrightarrow P_i y)$.

Assume first that a **CE**-valuation $\varphi_{\mathbf{CE}}$ of A is given such that $\varphi_{\mathbf{CE}}(A) = 1$. By the Löwenheim-Skolem's theorem we can assume the universe U of its model to be countable, without loss of generality. Therefore also the set of equivalence classes $[u]_i = \{v \mid \varphi_{\mathbf{CE}}(u \equiv_i v) = 1\}$ with respect to the two equivalence relations is countable. Since our Gödel set V is infinite, there exists a injection λ:

$$\lambda : \{[u]_1, [u]_2 : u \in U\} \to V \setminus \{0, 1\}.$$

Using this injection we define the valuation in $\mathbf{G}^{\triangle}_V$ as

$$\varphi_{\mathbf{G}}(P_i u) = \lambda([u]_i).$$

We now show that the assumptions of Lemma 1 are fulfilled:

$$\begin{aligned}\varphi_{\mathbf{CE}}(u^k_j \equiv_i v^k_j) = 1 \quad &\Leftrightarrow \quad [u^k_j]_i = [v^k_j]_i \quad \Leftrightarrow \quad \lambda([u^k_j]_i) = \lambda([v^k_j]_i)\\ &\Leftrightarrow \quad \varphi_{\mathbf{G}}(P_i u^k_j) = \varphi_{\mathbf{G}}(P_i u^k_j) \quad \Leftrightarrow \quad \varphi_{\mathbf{G}}(P_i u^k_j \leftrightarrow P_i u^k_j) = 1\\ &\Leftrightarrow \quad \varphi_{\mathbf{G}}(\triangle(P_i u^k_j \leftrightarrow P_i u^k_j)) = 1 \quad \Leftrightarrow \quad \varphi_{\mathbf{G}}(\sigma(u^k_j \equiv_i v^k_j)) = 1\end{aligned}$$

We can similary prove that $\varphi_{\mathbf{CE}}(u^k_j \not\equiv_i v^k_j) = 1 \quad \Leftrightarrow \quad \varphi_{\mathbf{G}}(\sigma(u^k_j \not\equiv_i v^k_j)) = 1$. Moreover, $\varphi_{\mathbf{G}}(\sigma(u^k_j \equiv_i v^k_j))$ and $\varphi_{\mathbf{G}}(\sigma(u^k_j \not\equiv_i v^k_j))$ are either 0 or 1. Hence, by Lemma 1, $\varphi_{\mathbf{G}}(\sigma(A)) = 1$.

Now assume that there exists a valuation $\varphi_{\mathbf{G}}$ in $\mathbf{G}^{\triangle}_V$ such that $\varphi_{\mathbf{G}}(\sigma(A)) = 1$. Define the valuation $\varphi_{\mathbf{CE}}$ in classical logic as

$$\varphi_{\mathbf{CE}}(u \equiv_i v) = 1 \text{ iff } \varphi_{\mathbf{G}}(\sigma(u \equiv_i v)) = 1$$

the above equivalence chains together with Lemma 1 give $\varphi_{\mathbf{CE}}(A) = 1$. □

Corollary 5. SAT^m *is undecidable in prenex or witnessed Gödel logics with $\triangle$, in presence of two predicate symbols.*

Proof. The very same proof as for Theorem 3 can be used. □

The undecidability result applies also to the monadic fragments of Łukasiewicz and product logics, two important formalizations of fuzzy logic [8]. For the former logic, Ragaz [14] proved the undecidability of the satisfiability problem in presence of at least four predicate symbols. This result was extended in [5] to product logic.

Corollary 6. *Satisfiability of monadic Łukasiewicz and product logic extended with $\triangle$ is undecidable, in presence of at least two predicate symbols.*

Proof. Follows by the embeddability of $\mathbf{G}^{\triangle}_{[0,1]}$ in Łukasiewicz and product logic extended with $\triangle$, cf. [5]. □

4.2 Infinite Gödel Sets with $|0|_{\mathrm{CB}} > 0$

We show the undecidability of $\mathrm{SAT}^m_{\mathbf{G}_V}$ where V is infinite, the truth value 0 has Cantor-Bendixon rank $|0|_{\mathrm{CB}} > 0$ in V (i.e., 0 is not isolated) and there is a (0-ry) predicate S which is always evaluated to a real between 0 and 1.

Theorem 4. $\mathrm{SAT}^m_{\mathbf{G}_V}$ *is undecidable in any Gödel logic* $\mathbf{G}_V$ *where* $|0|_{\mathrm{CB}} > 0$ *in presence of at least three predicates one of which is a constant S such that* $0 < \varphi_{\mathbf{G}}(S) < 1$ *for each valuation* $\varphi_{\mathbf{G}}$.

Proof. Proceed similarly to that of Theorem 3. Here we define the (local) translation σ as follows:

$$\sigma(x^k_j \equiv_i y^k_j) = (P_i x^k_j \leftrightarrow P_i y^k_j)$$
$$\sigma(x^k_j \not\equiv_i y^k_j) = (P_i x^k_j \leftrightarrow P_i y^k_j) \rightarrow S$$

and translate the **CE** formula A as $\tau(A) = \sigma(A) \wedge \forall x(P_1 x \prec S) \wedge \forall x(P_2 x \prec S)$.

Assume first that a valuation $\varphi_{\mathbf{CE}}$ of A in classical logic is given such that $\varphi_{\mathbf{CE}}(A) = 1$. Following the first steps of the proof of Theorem 3 we define

$$\varphi_{\mathbf{G}}(P_i u) = \lambda([u]_i)$$

but here the injection λ maps equivalence classes into truth values below $\varphi_{\mathbf{G}}(S)$:

$$\lambda : \{[u]_i, [u]_2 : u \in U\} \rightarrow V \cap (0, \varphi_{\mathbf{G}}(S)).$$

This can always be achieved as 0 is not isolated, thus, below any given point in V, in our case below $\varphi_{\mathbf{G}}(S)$, there are at least countably many truth values.

We now show that the remaining assumptions of Lemma 1 are fulfilled (by definition $\varphi_{\mathbf{G}}(\sigma(u^k_j \equiv_i v^k_j))$ and $\varphi_{\mathbf{G}}(\sigma(u^k_j \not\equiv_i v^k_j))$ are either 1 or below $\varphi_{\mathbf{G}}(S)$). Obviously $\varphi_{\mathbf{CE}}(u^k_j \equiv_i v^k_j) = 1$ iff $\varphi_{\mathbf{G}}(P_i u^k_j) = \varphi_{\mathbf{G}}(P_i v^k_j)$. Moreover

$$\begin{aligned}\varphi_{\mathbf{CE}}(u^k_j \not\equiv_i v^k_j) = 1 \quad &\Leftrightarrow \quad [u^k_j]_i \neq [v^k_j]_i \quad \Leftrightarrow^1 \quad \lambda([u^k_j]_i) \neq \lambda([v^k_j]_i)\\ \varphi_{\mathbf{G}}(P_i u^k_j) \neq \varphi_{\mathbf{G}}(P_i u^k_j) \quad &\Leftrightarrow \quad \varphi_{\mathbf{G}}(P_i u^k_j \leftrightarrow P_i u^k_j) \leq \varphi_{\mathbf{G}}(S)\\ \Leftrightarrow \quad \varphi_{\mathbf{G}}((P_i u^k_j \leftrightarrow P_i u^k_j) \rightarrow S)) = 1 \quad &\Leftrightarrow \quad \varphi_{\mathbf{G}}(\sigma(u^k_j \not\equiv_i v^k_j)) = 1\end{aligned}$$

(where ^{1} follows from the injectivity of λ). By Lemma 1, we have $\varphi_{\mathbf{G}}(\sigma(A)) = 1$ and by the definition of $\varphi_{\mathbf{G}}$, $\varphi_{\mathbf{G}}(\tau(A)) = 1$.

For the converse direction, assume to have a valuation $\varphi_{\mathbf{G}}$ such that $\varphi_{\mathbf{G}}(\tau(A)) = 1$. Define the valuation $\varphi_{\mathbf{CE}}$ in a straightforward way as

$$\varphi_{\mathbf{CE}}(u \equiv_i v) = 1 \text{ iff } \varphi_{\mathbf{G}}(\sigma(u \equiv_i v)) = 1.$$

Notice that by $\varphi_{\mathbf{G}}(\forall x(P_i x \prec S)) = 1$ follows that $\varphi_{\mathbf{G}}(P_i x) < \varphi_{\mathbf{G}}(S) < 1$ for all x. The equivalences above together with Lemma 1 give $\varphi_{\mathbf{CE}}(A) = 1$. □

4.3 Infinite Gödel Sets with $|0|_{\mathrm{CB}} \geq 2$

We show the undecidability of $\mathrm{SAT}^m_{\mathbf{G}_V}$ where V is infinite, 0 has Cantor-Bendixon rank $|0|_{\mathrm{CB}} \geq 2$ (i.e., 0 is limit point of limit points) and the language of $\mathbf{G}_V$ contains at least three predicate symbols. $\mathbf{G}_{[0,1]}$being a prominent example.

Theorem 5. $\mathrm{SAT}^m_{\mathbf{G}_V}$ *is undecidable when* $|0|_{\mathrm{CB}} \geq 2$ *in* V *and in presence of at least three predicate symbols.*

Proof. Proceed similarly to the proofs of Theorem 4. First notice that the central property used in this proof (cf. Lemma 1) is that there is a way to decide whether an evaluation is below 1 or it is 1. Indeed, for the reverse direction of the undecidability proof (i.e., given a satisfying Gödel interpretation $\varphi_{\mathbf{G}_V}$ we have to construct a satisfying interpretation in **CE**) we need to select an open interval in V strictly between 0 and 1, objects with valuations of the equivalence predicates P_i within that interval, and use them to define the equivalence classes in **CE**. In Theorem 4, the predicate constant is used to this purpose. Here a formula $\exists y \exists z(P_i z \prec P_i y \wedge \dots)$ would not be enough as our 'strictly less' relation $\prec$ collapses at 1 (see Section 2) and therefore we cannot be sure that $\varphi_{\mathbf{G}_V}(P_i y)$ and $\varphi_{\mathbf{G}_V}(P_i z)$ are choosen below 1. To overcome this problem we use the third predicate P whose valuation we force to be a decreasing sequence to 0. This can be expressed by $\neg\forall x Px \wedge \forall x \neg\neg Px$, as the first conjunct says that the infimum of all evaluations of P is 0, and the second that *every* valuation of P is bigger than 0. So we can say that either the valuation of Px is 1, or we can find y and z 'below' x (i.e., the valuations of Py and Pz are below the valuation of Px) defining the needed open interval. This leads to the formula $\forall x(Px \vee \exists y \exists z(Pz \prec Py \wedge Py \prec Px))$; on the other hand requiring the existence of objects w with $\varphi_{\mathbf{G}_V}(P_i w)$ within the interval can be simply expressed by $\exists w(Pz \prec P_1 w \prec Py \wedge Pz \prec P_2 w \prec Py)$.

Consider now the forward direction of the proof, i.e., constructing a satisfying valuation in $\mathbf{G}_V$ from a satisfying valuation in **CE**. The use of the formulas above complicates the matters as we have to deal with countably many $\varphi_{\mathbf{G}}(Px)$ for $x \in U_{\mathbf{G}}$ and below each of them open intervals (coming from Py and Pz), in which the equivalence classes are interpreted in parallel. This gives rise to the condition that the Cantor-Bendixon rank of 0 is at least 2. This 'parallel construction' also hints that in translating our formula A of **CE** into a formula $\tau(A)$ of $\mathbf{G}_V$ we will duplicate the universe $U_{\mathbf{CE}}$ for each interval so that we can faithfully embed the equivalence classes of **CE** into the respective intervals. This leads to the fact that in the translated formula $\tau(A)$ the quantifiers act over a much larger universe, as we have to duplicate the universe for each interval. To confine the valuations of elements of the 'correct' interval we add in the formula (1a) below disjuncts evaluating to 1 for objects outside the considered interval.

We are now ready to present the formal definition of the translation of our formula A in **CE** into $\mathbf{G}_V$ which uses the (not anymore local) translation $\sigma_{a,b}(A)$, where a and b define the interval $[b, a]$ in which the evaluation takes place:

$$\sigma_{a,b}(\forall rB) = \forall r(P_1 r \prec Pb \vee Pa \prec P_1 r \vee P_2 r \prec Pb \vee Pa \prec P_2 r \vee \sigma_{a,b}(B)) \quad (1a)$$

$$\sigma_{a,b}(\exists rB) = \exists r((Pb \prec P_1 r \prec Pa) \wedge (Pb \prec P_2 r \prec Pa) \wedge \sigma_{a,b}(B)) \quad (1b)$$

$$\sigma_{a,b}(\bigvee_j \bigwedge_k (r_j^k \equiv_i s_j^k)^l) = \bigvee_j \bigwedge_k \sigma((r_j^k \equiv_i s_j^k)^l) \quad (1c)$$

$$\sigma_{a,b}(r \equiv_i s) = (P_i r \leftrightarrow P_i s) \quad (1d)$$

$$\sigma_{a,b}(r \not\equiv_i s) = ((P_i r \leftrightarrow P_i s) \rightarrow Pa)) \quad (1e)$$

The translation $\tau(A)$ of the **CE** formula A is defined as:

$$\tau(A) = \neg\forall x Px \wedge \forall x \neg\neg Px \wedge \quad (2a)$$

$$\forall x(Px \vee \exists y \exists z[\quad (2b)$$

$$Pz \prec Py \wedge Py \prec Px \wedge \quad (2c)$$

$$\forall u(Pu \rightarrow Pz \vee Py \rightarrow Pu) \wedge \quad (2d)$$

$$\exists w(Pz \prec P_1 w \prec Py \wedge Pz \prec P_2 w \prec Py) \wedge \quad (2e)$$

$$\sigma_{y,z}(A)]) \quad (2f)$$

As in the proofs of Theorems 3 and 4 we assume first that $\varphi_{\mathbf{CE}}(A) = 1$ and we construct an interpretation $\varphi_{\mathbf{G}}$ of $\mathbf{G}_V$ in which $\varphi_{\mathbf{G}}(\tau(A)) = 1$. By the Löwenheim-Skolem's theorem we assume that the domain $U_{\mathbf{CE}}$ of $\varphi_{\mathbf{CE}}$ is countable. As mentioned above we duplicate the universe $U_{\mathbf{CE}}$ countably many times, and for good measure we throw in another countable set of objects (c_n) which we use to define the decreasing sequence Pc_n to 0. Thus, the universe of our valuation $\varphi_{\mathbf{G}}$ is defined as $U_{\mathbf{G}} = \{u^n : u \in U_{\mathbf{CE}}, n \in \mathbb{N}\} \cup \{c_n : n \in \mathbb{N}\}$, where all the u^n and c_n are different. Note that we add the index n as superscript to u to indicate copies of the elements $u \in U_{\mathbf{CE}}$.

The values of P under $\varphi_{\mathbf{G}}$ meet the following requirements: $\varphi_{\mathbf{G}}(Pc_n)$ satisfies (i) $\varphi_{\mathbf{G}}(Pc_{n+1}) < \varphi_{\mathbf{G}}(Pc_n)$, (ii) $\lim_{n\to\infty} \varphi_{\mathbf{G}}(Pc_n) = 0$, (iii) $\varphi_{\mathbf{G}}(Pc_n) \in V' \setminus \{0, 1\}$, where V' is the Cantor-Bendixon derivative of V, and (iv) $\varphi_{\mathbf{G}}(Pu^n) = 1$ for all n and $u \in U_{\mathbf{CE}}$. This definition makes sure that $\varphi_{\mathbf{G}}((2a)) = 1$.

From $|0|_{\mathrm{CB}} \geq 2$ and (iii) follow that below any given $\varphi_{\mathbf{G}}(Pc_n)$ there exist countably many disjoint open intervals, each containing countably many truth values. Define $f : \mathbb{N} \to \mathbb{N}$ and K_n such that: (a) f is strictly monotone increasing, and (b) the open interval $K_n = (\varphi_{\mathbf{G}}(Pc_{f(n)+1}), \varphi_{\mathbf{G}}(Pc_{f(n)}))$ contains countably many truth values. As a consequence of (a) and (b) we have (c) the intervals K_n are all disjoint.

Since there are at least countably many truth values in each K_n there is for any n an injection $\lambda_n : \{[u]_1, [u]_2 : u \in U_{\mathbf{CE}}\} \to K_n \cap V$. The valuation of $P_i(u^n)$ is then defined as

$$\varphi_{\mathbf{G}}(P_i(u^n)) = \lambda_n([u]_i).$$

This ensures that (2e) is satisfied. To complete the definition of the valuation of atomic formulas we set $\varphi_{\mathbf{G}}(P_i(c_n)) = 1$ for all n.

We will now show that $\varphi_{\mathbf{G}}(\tau(A)) = 1$. Consider the universal quantifier $\forall x$ in (2b) and pick up an arbitrary element x from $U_{\mathbf{G}}$. If $x = u^n$ then $\varphi_{\mathbf{G}}(Px) = 1$ (see (iv) above). Assume now that $x = c_n$. We choose $c_{f(n)}$ for y and $c_{f(n)+1}$

for z. (Remember that the interval K_n defined through the evaluations of P with these elements contains countably many truth values). It is easy to see that with these chosen y and z the parts (2c) and (2d) are satisfied.

To prove that $\varphi_{\mathbf{G}}(\tau(A)) = 1$ it remains to show that $\varphi_{\mathbf{G}}(\sigma_n(A)) = 1$ where σ_n is a shorthand for $\sigma_{c_{f(n)},c_{f(n)+1}}$. We can indeed give a selection function for the existentially quantified variables: If all the quantifiers in front of an existential quantifier are instantiated, simply drop all the super-scripts in u^n, consider the resulting assignment in **CE**, and use the object selected by the existential quantifier there, adding the index of the current interval in which we evaluate. Therefore the existential quantifiers are always evaluated in the current interval, and thus the first two conjuncts of (1b) are satisfied.

On the other hand considering the universal quantifier and (1a) we see that if the object instantiating the universal quantifier is outside the current interval, i.e., the valuations are outside the interval defined by $\varphi_{\mathbf{G}}(Py) = \varphi_{\mathbf{G}}(Pc_{f(n)})$ and $\varphi_{\mathbf{G}}(Pz) = \varphi_{\mathbf{G}}(Pc_{f(n)+1})$, the evaluation of (1a) immediately becomes 1.

So we can assume in the following that all objects instantiating quantified variables in (2f), i.e., in $\sigma_{y,z}(A)$, give valuations of P_1 and P_2 within the interval under discussion. The very same computations as in Theorem 4 (with $Pc_{f(n)}$ playing the role of S) show $\varphi_{\mathbf{G}}(\sigma_n(A)) = 1$ for each n if $\varphi_{\mathbf{CE}}(A) = 1$. Hence $\varphi_{\mathbf{G}}(\tau(A)) = 1$ if $\varphi_{\mathbf{CE}}(A) = 1$.

For the reverse direction assume that $\varphi_{\mathbf{G}}(\tau(A)) = 1$. For $\varphi_{\mathbf{G}}$ the following hold: (i) there exists an u' such that $0 < \varphi_{\mathbf{G}}(Pu') < 1$, as $\varphi_{\mathbf{G}}((2a)) = 1$; (ii) there are $y = v$ and $z = w$ such that (2c)-(2e) hold. Define the universe of $\varphi_{\mathbf{CE}}$ as

$$U_{\mathbf{CE}} = \{u \in U_{\mathbf{G}} : \varphi_{\mathbf{G}}(Pw) < \varphi_{\mathbf{G}}(P_i u) < \varphi_{\mathbf{G}}(Pv),\ i = 1, 2\}$$

Note that $U_{\mathbf{CE}}$ cannot be empty as $\varphi_{\mathbf{G}}((2e)) = 1$. Define a valuation $\varphi_{\mathbf{CE}}$ as

$$\varphi_{\mathbf{CE}}(a \equiv_i b) = 1 \quad \text{iff} \quad \varphi_{\mathbf{G}}(P_i a \leftrightarrow P_i b) = 1$$

from which follows that $\varphi_{\mathbf{CE}}(A) = 1$ being $\varphi_{\mathbf{CE}}(A)$ nothing but the valuation $\varphi_{\mathbf{G}}(\sigma_{v,w}(A))$. □

Notice that all infinite-valued Gödel logics $\mathbf{G}_V$ with at least three predicate symbols satisfy the hypothesis of the theorem above, with the exception of those in which $|0|_{\mathrm{CB}} = 0$ or $|0|_{\mathrm{CB}} = 1$ in V. In the former case, Theorem 1 ensures the decidability of $\mathrm{SAT}^m_{\mathbf{G}_V}$. We show below that the latter case, which is the only case left open, refers, in fact, to *one* Gödel logic: the one known as $\mathbf{G}_\downarrow$ and in which $V = \{1/n : n \in \mathbb{N}\} \cup \{0\}$.

Proposition 1. *If V has $|0|_{\mathrm{CB}} = 1$, then $\mathrm{SAT}_{\mathbf{G}_V}$ is equivalent to $\mathrm{SAT}_{\mathbf{G}_\downarrow}$.*

Proof. For any Gödel set V in which $|0|_{\mathrm{CB}} = 1$ there is a $\lambda \notin V$, $0 < \lambda < 1$, such that below λ there are only isolated truth values and 0. Using the same technique as in Theorem 1 but projecting everything above λ to 1 we see that all such V are order isomorphic to $\{1/n : n \in \mathbb{N}\} \cup \{0\}$; i.e., to the truth values set of $\mathbf{G}_\downarrow$. □

References

1. Baaz, M.: Infinite-valued Gödel logics with 0-1-projections and relativizations. In: Proceedings Gödel 1996. Kurt Gödel's Legacy. LNL, vol. 6, pp. 23–33. Springer, Heidelberg (1996)
2. Baaz, M., Ciabattoni, A., Fermüller, C.G.: Herbrand's Theorem for Prenex Gödel Logic and its Consequences for Theorem Proving. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001. LNCS, vol. 2250, pp. 201–216. Springer, Heidelberg (2001)
3. Baaz, M., Ciabattoni, A., Fermüller, C.G.: Monadic Fragments of Gödel Logics: Decidability and Undecidability Results. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 77–91. Springer, Heidelberg (2007)
4. Baaz, M., Preining, N., Zach, R.: First-order Gödel logics. Annals of Pure and Applied Logic 147, 23–47 (2007)
5. Baaz, M., Hájek, P., Svejda, D., Krajícek, J.: Embedding Logics into Product Logic. Studia Logica 61(1), 35–47 (1998)
6. Beckmann, A., Goldstern, M., Preining, N.: Continuous Fraïssé conjecture. Order 25(4), 281–298 (2008)
7. Gabbay, D.M.: Decidability of some intuitionistic predicate theories. J. of Symbolic Logic 37, 579–587 (1972)
8. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
9. Hájek, P.: Arithmetical complexity of fuzzy predicate logics: a survey II. Annals of Pure and Applied Logic (to appear)
10. Hájek, P.: Monadic Fuzzy Predicate Logics. Studia Logica 71(2), 165–175 (2002)
11. Hájek, P., Cintula, P.: On theories and models in fuzzy predicate logics. Journal of Symbolic Logic 71(3), 863–880 (2006)
12. Kechris, A.S.: Classical Descriptive Set Theory. Springer, Heidelberg (1995)
13. Preining, N.: Complete Recursive Axiomatizability of Gödel Logics. PhD thesis, Vienna University of Technology, Austria (2003)
14. Ragaz, M.: Die Unentscheidbarkeit der einstelligen unendlichwertigen Prädikatenlogik. Arch. math. Logik 23, 129–139 (1983)
15. Rogers, H.: Certain logical reduction and decision problems. Annals of Mathematics 64, 264–284 (1956)
16. Takeuti, G., Titani, T.: Intuitionistic fuzzy logic and intuitionistic fuzzy set theory. J. of Symbolic Logic 49, 851–866 (1984)

Learning by Questions and Answers: From Belief-Revision Cycles to Doxastic Fixed Points

Alexandru Baltag[1] and Sonja Smets[2,3]

[1] Computing Laboratory, University of Oxford
Alexandru.Baltag@comlab.ox.ac.uk
[2] Dep. of Artificial Intelligence & Dep. of Philosophy, University of Groningen
[3] IEG Research Group, University of Oxford
S.J.L.Smets@rug.nl

1 Introduction

We investigate the *long-term behavior of iterated belief revision with higher-level doxastic information*. While the classical literature on iterated belief revision [13, 11] deals only with *propositional* information, we are interested in *learning* (by an *introspective* agent, of some *partial* information about the) answers to various questions Q_1, $Q_2, \ldots$, $Q_n, \ldots$ that *may refer to the agent's own beliefs* (or even to her *belief-revision plans*). Here, "learning" can be taken either in the "hard" sense (of becoming absolutely certain of the answer) or in the "soft" sense (accepting some answers as more plausible than others). If the questions are *binary* ("is φ true or not?"), the agent "learns" a sequence of *true* doxastic sentences $\varphi_1, \ldots, \varphi_n, \ldots$. "Investigating the long-term behavior" of this process means that we are interested in whether or not the agent's beliefs, her "knowledge" and her conditional beliefs stabilize eventually or keep changing forever.

The initial beliefs are given by a finite "plausibility" structure as in [5, 7, 6]: a finite set of possible worlds with a total preorder, representing the worlds' *relative plausibility*. These structures are standard in Belief Revision: they are special cases of Halpern's "preferential models" [20], Grove's "sphere" models [19], Spohn's ordinal-ranked models [25] and Board's "belief-revision structures" [10]. A *(pointed plausibility) model* is a plausibility structure with a designated state (representing the *actual state* of the real world) and a valuation map (capturing the *ontic "facts"* in each possible state of the world). We adopt our *finiteness* assumption for three reasons. First, *all the above-mentioned semantic approaches are equivalent in the finite case*: this gives a certain "semantic *robustness*" to our results. Secondly, it seems *realistic* to assume that an agent starts with a belief base consisting of *finitely* many sentences (believed on their own merits, even if by logical omniscience, all their infinitely many consequences are also believed). Since all the languages considered here have the finite model property, any finite base can be represented in a finite model. Thirdly, our problem as formulated above makes sense primarily for finite structures: *only for finite models it is reasonable to expect that revision might stabilize after finitely many iterations.*

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 124–139, 2009.

A *belief upgrade* will be a type of *model transformation*, i.e. a (partial) function taking (pointed plausibility) models as inputs and returning "upgraded" models as outputs. Examples of upgrades have been proposed in the literature on Belief Revision e.g. by Boutilier [12] and Rott [24], in the literature on dynamic semantics for natural language by Veltman [26], and in Dynamic Epistemic Logic by J. van Benthem [9]. Three of them ("update", "radical" upgrade and "conservative" upgrade) are mentioned in this paper, and our results bear specific relevance to them. But we also generalize these proposals to a natural notion of "belief upgrade", in the spirit of Nayak [23], which turns out to be a special case of our *"Action Priority Update"* [7]. So our results have a general significance: they are applicable to any reasonable notion of iterated belief revision.

Not every model transformation makes sense as a way to change beliefs. The fact that we are dealing with *purely "doxastic" events* imposes some obvious restrictions: we assume nothing else happens in the world except for our agent's change of beliefs; so the upgrade should not change the facts of the world but only the doxastic state. Hence, *the actual state and the valuation of atomic sentences should stay the same* (since we interpret atomic sentences as *ontic, non-doxastic "facts"*). The only things that might change are *the set S of possible worlds and the order $\leq$*: the first reflects a change in the agent's knowledge, while the second reflects a change in her (conditional) beliefs. Moreover, in the *single-agent case*, it is enough to consider only changes in which the new set S' of states is *a subset $S' \subseteq S$* of the old set: since our agent has *perfect recall* and nothing else happens in the world except for her own belief change, *there is no increase in uncertainty*; hence, the set of possibilities can only shrink or stay the same[1].

So single-agent upgrades correspond essentially to a *shrinking (or maintaining) of the set of possible worlds, combined with a relational transformation.* Still, not even every transformation of this type will make sense as a doxastic change. Our idea is that, in general, *a belief upgrade comes by learning some (certain or uncertain) information about the answer to some specific question.*

There is a large literature on the semantics and pragmatics of questions, starting with the classical work of Grice [17]. Here, we adopt Groenendijk's approach to questions-as-partitions [18]. *Questions in a given language* are in this view *partitions of the state space*, such that each cell of the partition is *definable by a sentence in the language.* In particular, a binary question is given by a two-cell partition $\{A, \neg A\}$ definable by a sentence A, while a general question ("which of the following is true...?") may have more cells $\{A_1, \ldots, A_n\}$, each corresponding to a possible answer A_i. As underlying languages, we consider the language of *basic doxastic logic* (with operators for "belief") and *its extension with conditional belief operators*, as in [10, 9, 5, 7, 6]. In addition, we use *dynamic operators* for "learning" answers to questions: these operators are eliminable, but they provide a compact way of expressing properties of dynamic belief revision.

In the "hard" sense, the action of "learning the correct answer" to a question $Q = \{A_1, \ldots, A_n\}$ restricts the state space to the cell $A := A_i$ that contains

[1] As shown in the Dynamic Epistemic Logic literature, this is *not* the case for multi-agent upgrades, see e.g. [4, 3, 7, 6].

the real state of the world; the plausibility relation and the valuation are correspondingly restricted, while the real world is left the same. This operation (as a possible semantics for the AGM revision operator $T * A$) is known as *conditioning* in the literature on Belief Revision (as a qualitative analogue of Bayesian conditioning), and as *update* (or "public announcement") in the Dynamic Epistemic Logic literature [8]. Intuitively, it represents the kind of learning in which the agent becomes *absolutely certain* that the correct answer is A: she comes to "know" that A was true, in an *irrevocable, un-revisable* way. Such learning can *never be undone*: the deleted states can never be considered "possible" again.

However, one can also learn the answer to a question $Q = \{A_1, \ldots, A_n\}$ in a "softer" (and more realistic) sense of "learning": one may come to believe that the correct answer $A := A_i$ is *more plausible* than the others; in addition, one may come to *rank the plausibility of the various other possible answers*. This means that what is "learned" is in fact *a plausibility order on the cells of the partition* Q. The effect of this type of learning is that the order between states in different cells A_i, A_j $(i \neq j)$ is changed (in accordance to the newly learned plausibility relation on cells), while the order between states in the same cell is kept the same. This is the *obvious qualitative analogue of Jeffrey Conditioning*. An example is the *"radical" (or "lexicographic") upgrade* $\Uparrow A$ from [24, 9]: the A-worlds are "promoted", becoming more plausible than the non-A-worlds, while within the two zones the old order is kept the same. This is a soft learning of the answer to a binary question $Q = \{A, \neg A\}$. Our setting can also capture another type of doxastic transformation proposed in the literature: the *"conservative" upgrade* $\uparrow A$, promoting only the most plausible A-worlds (making them the most plausible overall), and leaving everything else the same.

In practice, we may *combine soft "Jeffreyan" learning with hard "Bayesian" learning*: while ranking some of the possible answers in their order of plausibility, one may simultaneously come to exclude other answers as impossible. In pragmatic terms: although receiving only soft information about the correct answer, one may also learn in the hard sense that the correct answer is *not* one of the answers to be excluded. So the most general notion of "learning the answer" to a question $Q = \{A_1, \ldots, A_n\}$ is an action that throws out (all the states lying in) some of the cells of the partition, and imposes some (new) plausibility order between (the states in) any two remaining cells A_i, A_j $(i \neq j)$, while leaving unchanged the order between the states in the same cell.

Any such action of "learning an answer" to a question $Q = \{A_1, \ldots, A_n\}$ can thus be encoded as *a plausibility order (or preorder) on some set* $\{A_{i_1}, \ldots, A_{i_k}\}$ *of possible answers* (a subset of the set of *all* possible answers $\{A_1, \ldots, A_n\}$). We call this type of actions *belief upgrades*: they represent our proposal for a general notion of "natural single-agent belief-revision event". One can easily recognize them as a special case of the "doxastic action models" (or "doxastic event models"), introduced (as ordinal-ranked models) in [2, 14] and (as qualitative plausibility models) in [7, 6]: namely, the special case in which *the preconditions of every two events are mutually disjoint*. But the important thing is the *meaning* of this representation. The *intended effect* (as described in the paragraph above)

of this action on any initial plausibility model is a kind of lexicographic refinement of the "old" total preorder by the new preorder. This can be recognized as a special case (the *single-agent, partitional case*) of the *Action Priority Rule* introduced in our previous work [7, 6] on multi-agent belief revision. While this move was seen there as the natural generalization to a belief-revision-friendly setting of the so-called "update product" from Dynamic Epistemic Logic [4, 3, 15], the special case considered here is even more closely related to mainstream work in Belief Revision. Indeed, it is a semantic counterpart of Nayak's "Jeffryzation" of belief revision [23], as expressed in the following quote: "(...) one of the morals we have learnt from Richard Jeffrey's classical work (...) is that *in belief updating, the representation of the belief state, the representation of the new information and the result of the update should be of the same type.*" ([23], page 361).

An upgrade is said to be *"correct"* if *the believed answer is true*: the actual state belongs to the "most plausible cell" (according to the newly learnt plausibility relation on possible answers). In the case of radical upgrades $\Uparrow A$, this means that the upgrade is "truthful", i.e. A is true in the real world.

We are interested in *iterated belief revision processes induced by sequences of correct upgrades*. We formalize this in our notion of *"correct" upgrade streams*. A particular case is the one of *repeated (correct) upgrades*: the same (partial information about the same) answer is repeatedly learnt in the same way, so the same transformation is iterated. The reason this is a non-superfluous repetition is that the learnt information may be *doxastic*: it may refer to the agent's own (conditional) beliefs, which are subject to change during the process. Hence, the truth value of an answer may change; as a consequence, it may be non-redundant to announce again the same answer to the same question! This phenomenon is well-known in the multi-agent case, see e.g. the Muddy Children Puzzle [16]. But, despite the existence of Moore sentences [22] (e.g. "p is the case, but you don't believe it!"), many authors tend to think that the problem becomes trivial in the *(introspective) single-agent* case: since the agent already knows what she believes and what not, it may appear that whenever she learns a new information, she can separate its ontic content from its (already known) doxastic part, and can thus simply learn a non-doxastic sentence. This is indeed the case for a *fixed* initial model, but in general the process of "eliminating the doxastic part" is highly complex and model-dependent: in different contexts, the same doxastic sentence may convey different information. And we cannot simply disregard such sentences: new information may come "packaged" in such a way that *it explicitly refers to the agent's beliefs* in order to *implicitly convey important new information about reality*, in a way that is highly context-dependent. An example is the announcement: *"You are wrong about p: whatever you believe about (whether or not) p is false!"* This announcement cannot be truthfully repeated, since (like a Moore sentence) it *becomes false after being learnt.*

Our counterexamples in this paper are of a similar nature, but with added subtleties, designed to keep the announcement truthful after it was learned. They decisively show that Introspection does not trivialize the problem of repeated upgrades: even when applied on an initial *finite* model, *a correct upgrade* (with the same true sentence) *may be repeated 'ad infinitum', without ever reaching*

a fixed point of the belief-revision process! Finite models may keep oscillating, changing cyclicly, and so do the agent's conditional beliefs.

However, we also have some *positive convergence results*: when iterating correct upgrades (starting on a finite initial model), *the agent's simple (non-conditional) beliefs* (as well as her knowledge) *will eventually stabilize*, reaching a fixed point. Moreover, if we apply *correct repeated upgrades* with (the same) new information *refering only to simple (non-conditional) beliefs* (i.e. we repeatedly revise with the same sentence in basic doxastic logic), then *the whole model always stabilizes* after a finite number of repeated upgrades: so in this case even the agent's conditional beliefs reach a fixed point.

The above *stabilization results form the main technical contribution* of this paper (besides the above-mentioned easy, but rather surprising and conceptually important, counterexamples); their proofs are non-trivial and are relegated to the Appendix. These results are applicable in particular to the important case of iterating (or repeating) *truthful "radical" upgrades* $\Uparrow A$. In contrast (as our second counterexample shows), *even the simple beliefs may never stabilize when repeating truthful "conservative" upgrades* $\uparrow A$. The reason for which the above result doesn't apply is that conservative upgrades, no matter how "truthful", can still be "deceiving" (i.e. fail to be "correct").

2 Questions and Upgrades on Preferential Models

A *(finite, single-agent) plausibility frame* is a structure $(S, \leq)$, consisting of a *finite set* S of "states" and a *total preorder* (i.e. a reflexive, transitive and "connected"[2] binary relation on S) $\leq \subseteq S \times S$. The usual reading of $s \leq t$ is that "state s is *at least as plausible* as state t". We write $s < t$ for the *"strict" plausibility relation* (s is *more plausible* than t): $s < t$ iff $s \leq t$ but $t \not\leq s$. Similarly, we write $s \cong t$ for the *"equi-plausibility"* (or *indifference*) relation (s and t are *equally plausible*): $s \cong t$ iff both $s \leq t$ and $t \leq s$. The "connectedness" assumption is not actually necessary for any of the results in this paper[3], but it is convenient to have, as it simplifies the definition of (conditional) beliefs. For infinite models, it is usually assumed that the relation $\leq$ is well-founded, but in the finite case (the only one of concern here) this condition is automatically satisfied.

A *(pointed) plausibility model* is a structure $\mathbf{S} = (S, \leq, \|\cdot\|, s_0)$, consisting of a plausibility frame $(S, \leq)$ together with a *valuation map* $\|\cdot\| : \Phi \to \mathcal{P}(S)$, mapping every element p of some given set Φ of "atomic sentences" into a set of states $\|p\| \subseteq S$, and together with a *designated state* $s_0 \in S$, called the "actual state".

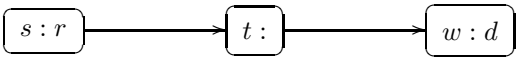
Knowledge, Belief and Conditional Belief. Given a plausibility model $\mathbf{S}$, and sets $P, R \subseteq S$, we put: $best\, P = Min_{\leq} P := \{s \in P : s \leq s' \text{ for all } s' \in P\}$, $best := best S$, $KP := \{s \in S : P = S\}$, $BP := \{s \in S : best \subseteq P\}$, $B^R P := \{s \in S : best\, R \subseteq P\}$.

[2] A binary relation $R \subseteq S \times S$ is called "connected" (or "complete") if for all $s, t \in S$ we have either sRt or tRs. A connected (pre)order is usually called a "total" (pre)order.

[3] And in fact for multi-agent frames it needs to be replaced by "local connectedness", see [7, 6].

Interpretation. The states of S represent *possible descriptions of the real world*. The *correct* description of the world is given by the "actual state" s_0. Intuitively, our (implicit) agent considers a description s as "possible" iff $s \in S$. The atomic sentences $p \in \Phi$ represent *"ontic" (i.e. non-doxastic) facts*. The valuation tells us which facts hold in which worlds. The plausibility relations $\leq$ capture the agent's *(conditional) beliefs* about the world: if the agent learnt that the real state is either s or t, she would believe (conditional on this information) that it was the *most plausible* of the two. *best* P is the set of "most plausible" states satisfying P, while KP, BP, $B^R P$ represent respectively the propositions "P is known", "P is believed" and "P is believed conditional on R". So *'knowledge"* corresponds to *truth in all the worlds that the agent considers as "possible"*. *Belief* corresponds to *truth in all the "most plausible" worlds*; while conditional belief given a condition R corresponds to truth in all the most plausible R-worlds. So *conditional beliefs* B^R *describe the agent's contingency plans for belief revision in case she learns* R. To quote J. van Benthem [9], conditional beliefs *"pre-encode" the agent's potential belief changes*. Note that the sets KP, BP, $B^R P$ *are always equal either to the empty set* $\emptyset$ *or to the whole state space* S; this reflects our implicit *Introspection* assumption: the agent knows what she knows (and what she believes) and what she doesn't know (or doesn't believe).

Example 1. Consider a pollster (Charles) with the following beliefs about how a voter (Mary) will vote:



In the representation, we skip the loops and the arrows that can be obtained by transitivity. There are three possible worlds s (in which Mary votes Republican), w (in which she votes Democrat) and t (in which she doesn't vote). We assume there are no other options: i.e. there are no other candidates and it is impossible to vote for both candidates. The atomic sentences are r (for "voting Republican") and d (for "voting Democrat"). The valuation is given in the diagram: $\|r\| = \{s\}$, $\|d\| = \{w\}$. *We assume the real world is* s, so in fact Mary will vote Republican! But Charles believes that she will vote Democrat (d); and in case this turns out wrong, he'd rather believe that she won't vote ($\neg d \wedge \neg r$) than accepting that she may vote Republican: so t is more plausible than s.

Doxastic Propositions. A *doxastic proposition* is a map $\mathbf{P}$ assigning to each model $\mathbf{S}$ some set $\mathbf{P_S} \subseteq S$ of states in S. We write $s \models_{\mathbf{S}} \mathbf{P}$, and say that $\mathbf{P}$ is *true at* s *in model* $\mathbf{S}$, iff $s \in \mathbf{P_S}$. We skip the subscript and write $s \models \mathbf{P}$ when the model is fixed. We say that $\mathbf{P}$ is *true in the (pointed) model* $\mathbf{S}$, and write $\mathbf{S} \models \mathbf{P}$, if it is true at the "actual state" s_0 in the model $\mathbf{S}$ (i.e. $s_0 \models_{\mathbf{S}} \mathbf{P}$). We denote by Prop the family of all doxastic propositions. In particular, the "always true" $\top$ and "always false" $\bot$ propositions are given pointwise by $\bot_{\mathbf{S}} := \emptyset, \top_{\mathbf{S}} := S$. For each atomic sentence p there exists a corresponding proposition $\mathbf{p}$, given by $\mathbf{p_S} = \|p\|_{\mathbf{S}}$. All the Boolean operations on sets can be *lifted* pointwise to operations on Prop: negation $(\neg \mathbf{P})_{\mathbf{S}} := S \setminus \mathbf{P_S}$, conjunction $(\mathbf{P} \wedge \mathbf{R})_{\mathbf{S}} := \mathbf{P_S} \cap \mathbf{R_S}$ etc. Similarly, we can define pointwise the *"best" operator*, a special doxastic proposition **best**

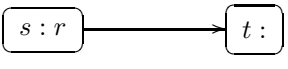
and the *epistemic and (conditional) doxastic modalities*: $(best\,\mathbf{P})_{\mathbf{S}} := best\,(\mathbf{P}_{\mathbf{S}})$, $\mathbf{best_S} := best\,\mathbf{S}$, $(K\mathbf{P})_{\mathbf{S}} := K\mathbf{P}_{\mathbf{S}}$ etc. Finally, the relation of *entailment* $\mathbf{P} \models \mathbf{R}$ is given pointwise by set-inclusion: $\mathbf{P} \models \mathbf{R}$ iff $\mathbf{P_S} \subseteq \mathbf{R_S}$ for all $\mathbf{S}$.

Questions and Answers. A *question* is a (finite) family $\mathbf{Q} = \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ of *exhaustive* and *mutually disjoint* propositions, i.e. $\bigvee_{i=1,n} \mathbf{A}^i = \top$ and $\mathbf{A}^i \wedge \mathbf{A}^j = \bot$ for all $i \neq j$. Every question $\mathbf{Q}$ induces a *partition* $\{\mathbf{A}^1_{\mathbf{S}}, \ldots, \mathbf{A}^n_{\mathbf{S}}\}$ on any model $\mathbf{S}$. Any of the sets $\mathbf{A}_i$ is a possible *answer* to question $\mathbf{Q}$. The *correct answer* to $\mathbf{Q}$ in a pointed model $\mathbf{S}$ is the unique *true* answer $\mathbf{A}^i$ (i.e. such $\mathbf{S} \models \mathbf{A}^i$).

Example 1, continued: In the above example, the relevant question is "how will Mary vote?" This can be represented as a question $\mathbf{Q} = \{\mathbf{r}, \mathbf{d}, \neg\mathbf{r} \wedge \neg\mathbf{d}\}$. Another relevant question is "will Mary vote Democrat?", given by $\mathbf{Q}' = \{\mathbf{d}, \neg\mathbf{d}\}$.

Learning the Answer with Certainty: Updates. The action by which the agent *learns with absolute certainty the correct answer* $\mathbf{A} = \mathbf{A}^i$ to a question $\mathbf{Q}$ is usually denoted by $!\mathbf{A}$. This operation is known as *'update"* in Dynamic Epistemic Logic, and as "*conditioning*" in the Belief Revision literature (where the term "update" is used for a different operation, the Katzuno-Mendelzon update [21]). This action *deletes all the non-*$\mathbf{A}$*-worlds from the pointed model, while leaving everything else the same.* The update $!\mathbf{A}$ is *executable* on a pointed model $\mathbf{S}$ *iff it is truthful*, i.e. $\mathbf{S} \models \mathbf{A}$. So formally, an update $!\mathbf{A}$ is a partial function that takes as input any pointed model $\mathbf{S} = (S, \leq, \| \|, s_0)$ satisfying $\mathbf{A}$ and returns a new pointed model $!\mathbf{A}(\mathbf{S}) = (S', \leq', \| \|', s'_0)$, given by: $S' = \mathbf{A_S}$, $s \leq' t$ iff $s \leq t$ and $s, t \in S'$, $\|p\|' = \|p\| \cap S'$, for all atomic p, and $s'_0 = s_0$.

Example 2. Consider the case in which Charles *learns for sure that Mary will not vote Democrat*: this is the update $!(\neg\mathbf{d})$, whose result is the updated model



Learning uncertain information: Upgrades. If the agent only *learns some uncertain information about the answer* to a question $\mathbf{Q}$, then what is actually learnt is *a plausibility relation* $\leq$ *on a subset* $\mathcal{A} \subseteq \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ *of the set of all possible answers.* Intuitively, the agent learns that the correct answer belongs to $\mathcal{A}$, and that some answers are more plausible than others. We encode this as a plausibility frame $(\mathcal{A}, \leq)$, called a *belief "upgrade", whose "worlds"* $\mathbf{A} \in \mathcal{A}$ *are mutually disjoint propositions.* This is a special case of "event models" [4, 3, 2, 14, 7, 6] (in which *every two events have mutually disjoint preconditions*).

A belief upgrade $\alpha = (\mathcal{A}, \leq)$ is *executable* on a model $\mathbf{S}$ if the disjunction of all its answers is true ($\mathbf{S} \models \bigvee \mathcal{A}$). *As an operation on (pointed) models*, the upgrade $\alpha = (\mathcal{A}, \leq)$ is *formally defined* to be a (partial) function α taking as input any model $\mathbf{S} = (S, \leq, \| \|, s_0)$ satisfying $\bigvee \mathcal{A}$, and returning a new model

$$\alpha(\mathbf{S}) = (S', \leq', \| \|', s'_0),$$

given by: $S' = (\bigvee \mathcal{A})_{\mathbf{S}}$; $s \leq' t$ iff either $s \in \mathbf{A}^i, t \in \mathbf{A}^j$ for some answers such that $\mathbf{A}^i < \mathbf{A}^j$, or else $s \leq t$ and $s, t \in \mathbf{A}^i$ for the same answer $\mathbf{A}^i$; $\|p\|' = \|p\| \cap S'$,

for all atomic p; and $s'_0 = s_0$. So this operation *deletes all the worlds not satisfying any of the answers in the list $\mathcal{A}$, and "promotes" the worlds satisfying more plausible answers, making them more plausible than the worlds satisfying less plausible answers*; in rest, *everything else stays the same.*

It is easy to see that the model $\alpha(\mathbf{S})$ obtained by applying a belief upgrade $\alpha = (\mathcal{A}, \leq)$ to an initial model $\mathbf{S}$ is precisely the *anti-lexicographic product "update"* $\mathbf{S} \otimes \alpha$ of the state model $\mathbf{S}$ and the event model $(\mathcal{A}, \leq)$, as prescribed by the *Action Priority Rule* in [7, 6].

Important Special Case: Standard Upgrades. An upgrade $(\mathcal{A}, \leq)$ is *"standard"* if the preorder $\leq$ is actually an *order*; in other words, if the plausibility relation between any two distinct answers $\mathbf{A}^i \neq \mathbf{A}^j$ in $\mathcal{A}$ is *strict* (i.e. $\mathbf{A}^i \not\geqslant \mathbf{A}^j$).

Lemma 1. Every belief upgrade is equivalent to a standard upgrade: for every upgrade α there is a standard upgrade α' s. t. $\alpha(\mathbf{S}) = \alpha'(\mathbf{S})$ for all models $\mathbf{S}$.

Standard Form. So we *can assume* (and from now we *will assume*) that *all our upgrades are standard.* This allows us to denote them using a special notation, called *standard form*: $(\mathbf{A}^1, \ldots, \mathbf{A}^n)$ represents the upgrade having $\{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ as its set of answers and in which the order is $\mathbf{A}^1 < \mathbf{A}^2 < \ldots < \mathbf{A}^n$.

Standard Upgrade Induced by a Propositional Sequence. Any sequence $\mathbf{P}^1, \ldots, \mathbf{P}^n$ of propositions induces a standard upgrade $[\mathbf{P}^1, \ldots, \mathbf{P}^n]$, given by

$$[\mathbf{P}^1, \ldots, \mathbf{P}^n] := (\mathbf{P}^1, \neg\mathbf{P}^1 \wedge \mathbf{P}^2, \ldots, \bigwedge_{1 \leq i \leq n-1} \neg\mathbf{P}_i \wedge \mathbf{P}^n).$$

Special Cases: updates, radical upgrades, conservative upgrades.
(1) Updates: $!\mathbf{P}$ is a special case of upgrade: $!\mathbf{P} = [\mathbf{P}] = (\mathbf{P})$.

(2) Radical Upgrades: An operation $\Uparrow \mathbf{P}$ called *"radical upgrade"* (or "lexicographic upgrade") was introduced by various authors [23], [24], [9], as one of the natural ways to give a "dynamic" semantics to the AGM belief revision operator [1]. This operation can be described as *"promoting" all the $\mathbf{P}$-worlds so that they become "better" (more plausible) than all $\neg\mathbf{P}$-worlds*, while *keeping everything else the same*: the valuation, the actual world and the relative ordering between worlds within either of the two zones ($\mathbf{P}$ and $\neg\mathbf{P}$) stay the same. Radical upgrade is a special case of upgrade:

$$\Uparrow \mathbf{P} = [\mathbf{P}, \top] = [\mathbf{P}, \neg\mathbf{P}] = (\mathbf{P}, \neg\mathbf{P}).$$

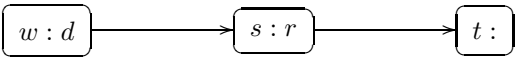
(3) Conservative Upgrades: Following [12, 24, 9], consider another operation $\uparrow \mathbf{P}$, called *"conservative upgrade"*. This corresponds to *"promoting" only the "best" (most plausible) $\mathbf{P}$-worlds, so that they become the most plausible overall*, while *keeping everything else the same.* This operation seems a more appropriate semantics for AGM revision, since it performs the "minimal" model-revision that is forced by believing $\mathbf{P}$. It is also a special case of our "upgrades":

$$\uparrow \mathbf{P} = \Uparrow (\mathit{best}\,\mathbf{P}) = [\mathit{best}\mathbf{P}, \top] = (\mathit{best}\,\mathbf{P}, \neg\mathit{best}\,\mathbf{P}).$$

Correctness. A standard upgrade $(\mathbf{A}^1, \ldots, \mathbf{A}^n)$ is said to be *correct* with respect to a pointed model $\mathbf{S}$ if *its most plausible answer is correct*; i.e. if $\mathbf{A}^1$ is true in the actual state: $\mathbf{S} \models \mathbf{A}^1$. A correct upgrade induces a *true belief* with respect to its underlying question: the agent comes to believe the correct answer.

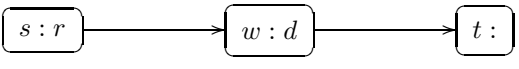
Correctness and Truthfulness. An update $!\mathbf{P}$, a radical upgrade $\Uparrow \mathbf{P}$ or a conservative upgrade $\uparrow \mathbf{P}$ are said to be *truthful* in a pointed model $\mathbf{S}$ if $\mathbf{S} \models \mathbf{P}$. It is easy to see that an *update* $!\mathbf{P}$, seen as an upgrade $(\mathbf{P})$, *is correct* (with respect to $\mathbf{S}$) *iff it is truthful*. Similarly, a *radical upgrade* $\Uparrow \mathbf{P}$, seen as an upgrade $(\mathbf{P}, \neg\mathbf{P})$, *is correct iff it is truthful*. However, for *conservative* upgrades, the two notions differ: seen as an upgrade $(\mathit{best}\,\mathbf{P}, \neg \mathit{best}\,\mathbf{P})$, $\uparrow \mathbf{P}$ is correct iff $\mathbf{S} \models \mathit{best}\,\mathbf{P}$ (which is not equivalent to $\mathbf{S} \models \mathbf{P}$). As a result, *correctness of a conservative upgrade implies truthfulness*, but *the converse is false*. So a *truthful conservative upgrade can still be "deceiving"* (i.e. incorrect)!

Examples. In Example 1, suppose that a *trusted informer* tells Charles that Mary will *not* vote Democrat. Charles is aware that the informer is *not infallible*: so this is not an update! Nevertheless, if Charles has a *strong trust* in the informer, then this learning event is a *radical upgrade* $\Uparrow(\neg\mathbf{d})$, whose result is:



Now, Charles believes that Mary will not vote at all (and so that she won't vote Democrat)! But if he later learned that this is not the case, he'd still keep his new belief that she won't vote Democrat (so he'd conclude she votes Republican).

Contrast this situation with the case in which Charles *hears a rumor* that Mary will not vote Democrat. Charles *believes* the rumor, but only *barely*: if he later needs to revise his beliefs, he'd give up immediately his belief in the rumor. This is a *conservative upgrade* $\uparrow \neg\mathbf{d}$ of the model in Example 1, resulting in:



Now, Charles believes Mary will not vote, but if he later learned this was not the case, he'd dismiss the rumor and revert to believing that Mary votes Democrat.

Operations on Upgrades: Dynamic Modalities and Sequential Composition. Given an upgrade α, we can define a *dynamic upgrade modality* $[\alpha]$, associating to any doxastic proposition $\mathbf{P}$ another proposition $[\alpha]\mathbf{P}$, given by

$$([\alpha]\mathbf{P})_{\mathbf{S}} := \mathbf{P}_{\alpha(\mathbf{S})}.$$

Since upgrades are functions, we can *compose* them functionally $\alpha \circ \alpha'$ or relationally $\alpha; \alpha'$ $(=\alpha' \circ \alpha)$. The resulting function is again an upgrade:

Proposition 2. Upgrades are closed under composition.

The Logic of Upgrades and Conditional Beliefs. We can use the standard form to introduce a nice syntax for the logic of upgrades and conditional beliefs:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \wedge \varphi \mid B^{\varphi}\varphi \mid [[\varphi, \ldots, \varphi]]\varphi$$

Here $[[\varphi^1, \ldots, \varphi^n]]$ denotes the dynamic modality for the standard upgrade $[\varphi^1, \ldots, \varphi^n]$. This logic can be embedded in the logic in [7, 6]. The semantics is defined compositionally, in the obvious way: to each sentence φ we associate a doxastic proposition $\|\varphi\|$, by induction on the structure of φ, e.g. $\|B^\varphi\psi\| = B^{\|\varphi\|}\|\psi\|$ etc. The operators K, $[!\varphi]$ and $[\Uparrow \varphi]$ can be *defined* in this logic. There exists a *complete axiomatization* (to be presented in an extended version of this paper).

3 Iterated Upgrades

Since in this paper we are concerned with the problem of *long-term learning via iterated belief revision*, we need to look at *sequences of upgrades*. We are particularly interested in learning *true* information, and so in iterating *correct upgrades*. Our main problem is *whether or not the iterated revision process induced by correct upgrades converges to a stable set of (conditional) beliefs*.

An *upgrade stream* $\boldsymbol{\alpha}$ is an infinite sequence of upgrades $(\alpha_n)_{n\in N}$. An upgrade stream is *definable in a logic* L if for every n all the answers of α_n are propositions that are definable in L. A *repeated upgrade* is an upgrade stream of the form $(\alpha, \alpha, \ldots)$, i.e. one in which $\alpha_n = \alpha$ for all $n \in N$.

Any upgrade stream induces a function mapping every model $\mathbf{S}$ *into a (possibly infinite) sequence* $\boldsymbol{\alpha}(\mathbf{S}) = (\mathbf{S}_n)$ *of models*, defined inductively by:

$$\mathbf{S}_0 = \mathbf{S}, \quad \text{and} \quad \mathbf{S}_{n+1} = \alpha_n(\mathbf{S}_n), \text{ if } \alpha_n \text{ is executable on } \mathbf{S}_n.$$

The upgrade stream $\boldsymbol{\alpha}$ is said to be *executable* on a pointed model $\mathbf{S}$ if every α_n is executable on $\mathbf{S}_n$ (for all $n \in N$). Similarly, $\boldsymbol{\alpha}$ is said to be *correct* with respect to $\mathbf{S}$ if α_n is correct with respect to $\mathbf{S}_n$, for every $n \in N$. If an upgrade stream is an *update stream* (i.e. one consisting only of updates), or a *radical upgrade stream* (consisting only of radical upgrades), or a *conservative upgrade stream*, then we call it *truthful* if every α_n is truthful with respect to $\mathbf{S}_n$.

We say that an upgrade stream $\boldsymbol{\alpha}$ *stabilizes a (pointed) model* $\mathbf{S}$ if the process of model-changing induced by $\boldsymbol{\alpha}$ reaches a fixed point; i.e. there exists some $n \in N$ such that $\mathbf{S}_n = \mathbf{S}_m$ for all $m > n$.

We say $\boldsymbol{\alpha}$ *stabilizes the agent's (simple, non-conditional) beliefs on the model* $\mathbf{S}$ if the process of belief-changing induced by $\boldsymbol{\alpha}$ on $\mathbf{S}$ reaches a fixed point; i.e. if there exists some $n \in N$ such that $\mathbf{S}_n \models B\mathbf{P}$ iff $\mathbf{S}_m \models B\mathbf{P}$, for all $m > n$ and all doxastic propositions $\mathbf{P}$. Equivalently, iff there exists some $n \in N$ such that $best_{\mathbf{S}_n} = best_{\mathbf{S}_m}$ for all $m > n$.

Similarly, we say $\boldsymbol{\alpha}$ *stabilizes the agent's conditional beliefs on the model* $\mathbf{S}$ if the process of conditional-belief-changing induced by $\boldsymbol{\alpha}$ on $\mathbf{S}$ reaches a fixed point; i.e. if there exists $n \in N$ such that $\mathbf{S}_n \models B^{\mathbf{R}}\mathbf{P}$ iff $\mathbf{S}_m \models B^{\mathbf{R}}\mathbf{P}$, for all $m > n$ and all doxastic propositions $\mathbf{P}, \mathbf{R}$. Equivalently, iff there exists $n \in N$ such that $(best\mathbf{P})_{\mathbf{S}_n} = (best\mathbf{P})_{\mathbf{S}_m}$ for all $m > n$ and all doxastic propositions $\mathbf{P}$.

Finally, $\boldsymbol{\alpha}$ *stabilizes the agent's knowledge on the model* $\mathbf{S}$ if the knowledge-changing process induced by $\boldsymbol{\alpha}$ on $\mathbf{S}$ reaches a fixed point; i.e. if there exists $n \in N$ such that $\mathbf{S}_n \models K\mathbf{P}$ iff $\mathbf{S}_m \models K\mathbf{P}$ for all $m > n$ and all propositions $\mathbf{P}$. Equivalently, iff there exists $n \in N$ such that $S_n = S_m$ for all $m > n$.

Lemma 3. An upgrade stream stabilizes a pointed model iff it stabilizes the agent's conditional beliefs on that model.

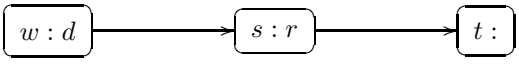
Lemma 4. If an upgrade stream stabilizes the agent's conditional beliefs then it also stabilizes her knowledge and her (simple) beliefs.

Proposition 5. Every upgrade stream stabilizes the agent's *knowledge*.

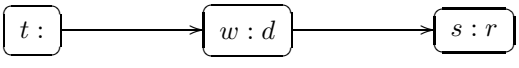
Corollary 6. [8] Every *executable update* stream stabilizes the model (and thus it stabilizes the agent's knowledge, beliefs and conditional beliefs).

The analogue of Corollary 6 is *not true* for arbitrary upgrade streams, *not even for correct upgrade streams. It is not even true for repeated correct upgrades*:

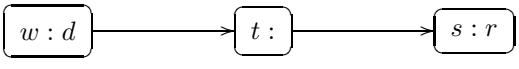
Counterexample: In Example 1, suppose that the trusted informer tells Charles the following true statement **A**: "*If you'd truthfully learn that Marry won't vote Republican, then your resulting belief about whether or not she votes Democrat would be wrong*". So **A** is the sentence $\mathbf{r} \vee (\mathbf{d} \wedge \neg B^{\neg \mathbf{r}}\mathbf{d}) \vee (\neg \mathbf{d} \wedge B^{\neg \mathbf{r}}\mathbf{d})$. This radical upgrade $\Uparrow \mathbf{A}$ is *truthful* (since **A** is true in s), and so *correct*, yielding



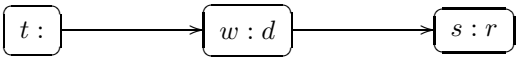
In this model, **A** is true in s (and in w), so another correct upgrade $\Uparrow \mathbf{A}$ yields:



Yet another correct upgrade with the same proposition produces



then another correct upgrade $\Uparrow \mathbf{A}$ *gets us back to the previous model*:



Clearly from now on the last two models *will keep reappearing, in an endless cycle*: hence in this example, *conditional beliefs never stabilize*! But note that *the simple beliefs are the same* in the last two models: so *the set of simple (non-conditional) beliefs stays the same from now on.* This is not an accident, but a symptom of a more general converge phenomenon:

Theorem 7. Every *correct* upgrade stream *stabilizes the agent's beliefs.*

This is the *main result* of this paper (proved in the *Appendix*), and it has a number of important consequences.

Corollary 8. Every *correct repeated upgrade* $\alpha, \alpha, \ldots, \alpha, \ldots$ whose answers are *definable in doxastic-epistemic logic* (i.e. the language of simple belief and knowledge operators) *stabilizes every model*, and thus *stabilizes the conditional beliefs.*

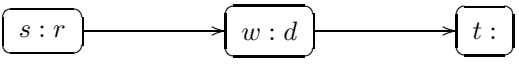
Corollary 9. Every *truthful radical upgrade stream* $(\Uparrow \mathbf{P}_i)_{i \in N}$ stabilizes the agent's *beliefs.*

Corollary 10. Every *repeated stream of truthful radical upgrades* $\Uparrow \varphi, \ldots, \Uparrow \varphi$ of a sentence *definable in doxastic-epistemic logic stabilizes every model* (with

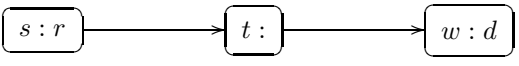
respect to which it is truthful), and thus *stabilizes the agent's conditional beliefs.*

The (analogues of the) last two Corollaries *fail for conservative upgrades*:

Counterexample: In the situation **S** from Example 1, suppose Charles hears a rumor **B** saying that: "*Either Mary will vote Republican or else your beliefs about whether or not she votes Democrat are wrong*". So **B** is the sentence $\mathbf{r} \vee (\mathbf{d} \wedge \neg B\mathbf{d}) \vee (\neg\mathbf{d} \wedge B\mathbf{d})$. The conservative upgrade $\uparrow \mathbf{B}$ is *truthful*: $s \in \mathbf{B_S}$. (But this is *not a correct upgrade*: $s \notin (best\, \mathbf{B})_{\mathbf{S}} = \{t\}$.) The result of the upgrade is:



Again, the sentence **B** is true in the actual state s (as well as in w), so the rumor $\uparrow \mathbf{B}$ can again be truthfully spread, resulting in... the original model once again:



So from now on, the two models (that support *opposite beliefs*!) *will keep reappearing, in an endless cycle*: the agent's beliefs will never stabilize!

4 Conclusions

This paper is an investigation of the long-term behavior of iterated belief revision with higher-level doxastic information. We propose a general notion of "correct belief upgrade", based on the idea that *non-deceiving belief revision is induced by learning (partial, but true information about) answers to questions.* The surprising conclusion of our investigation is that *iterated revision with doxastic information is highly non-trivial, even in the single-agent case.* More concretely, *the revision is not guaranteed to reach a fixed point*, even when indefinitely repeating the same correct upgrade: *neither the models, nor the conditional beliefs are necessarily stabilized.* But we also have some important *stabilization results*: *both knowledge and beliefs are eventually stabilized by iterated correct upgrades*; moreover, *both the models and the conditional beliefs are stabilized by repeated correct upgrades with non-conditional information (expressible in doxastic-epistemic logic).* These positive results have a wide scope, being applicable in particular to truthful radical upgrades (as well as to "updates"). However, *their range of applicability is limited by the existence of "truthful-but-deceiving" (incorrect) upgrades*: in particular, the repetition of the conservative upgrade in our last counterexample leads to infinite cycles of non-stabilizing beliefs.

Acknowledgments. The authors give special thanks to J. van Benthem for stimulating discussions and seminal ideas. We thank D. Mackinson for his insightful commentary on the second's author's LSE presentation of preliminary work leading to this paper. We thank to an anonymous WOLLIC referee for useful references. The research of the first author was partially supported by the Netherlands Organization for Scientific Research, grant number B 62-635, which is herewith gratefully acknowledged. The second author acknowledges the support by the University of Groningen via a Rosalind Franklin research position.

References

[1] Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. JSL 50, 510–530 (1985)

[2] Aucher, G.: A combined system for update logic and belief revision. Master's thesis, ILLC, University of Amsterdam, Amsterdam, the Netherlands (2003)

[3] Baltag, A., Moss, L.S.: Logics for epistemic programs. Synthese 139, 165–224 (2004); Knowledge, Rationality & Action 1–60

[4] Baltag, A., Moss, L.S., Solecki, S.: The logic of common knowledge, public announcements, and private suspicions. In: Gilboa, I. (ed.) Proc. of TARK 1998, pp. 43–56 (1998)

[5] Baltag, A., Smets, S.: Conditional doxastic models: a qualitative approach to dynamic belief revision. ENTCS 165, 5–21 (2006)

[6] Baltag, A., Smets, S.: The logic of conditional doxastic actions. Texts in Logic and Games 4, 9–32 (2008)

[7] Baltag, A., Smets, S.: A qualitative theory of dynamic interactive belief revision. Texts in Logic and Games 3, 9–58 (2008)

[8] van Benthem, J.F.A.K.: One is a lonely number. In: Koepke, P., Chatzidakis, Z., Pohlers, W. (eds.) Logic Colloquium 2002, pp. 96–129. ASL and A.K. Peter, Wellesley (2006)

[9] van Benthem, J.F.A.K.: Dynamic logic of belief revision. JANCL 17(2), 129–155 (2007)

[10] Board, O.: Dynamic interactive epistemology. Games and Economic Behaviour 49, 49–80 (2002)

[11] Booth, R., Meyer, T.: Admissible and restrained revision. Journal of Artificial Intelligence Research 26, 127–151 (2006)

[12] Boutilier, C.: Iterated revision and minimal change of conditional beliefs. JPL 25(3), 262–305 (1996)

[13] Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artificial Intelligence 89, 1–29 (1997)

[14] van Ditmarsch, H.P.: Prolegomena to dynamic logic for belief revision. Synthese 147, 229–275 (2005)

[15] van Ditmarsch, H.P., van der Hoek, W., Kooi, B.P.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2007)

[16] Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)

[17] Grice, P.: Logic and conversation. In: Studies in the Ways of Words. Harvard University Press, Cambridge (1989)

[18] Groenendijk, J., Stokhof, M.: Questions. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, pp. 1055–1124. Elsevier, Amsterdam (1997)

[19] Grove, A.: Two modellings for theory change. JPL 17, 157–170 (1988)

[20] Halpern, J.Y.: Reasoning about Uncertainty. MIT Press, Cambridge (2003)

[21] Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. Cambridge Tracts in Theoretical Computer Science, pp. 183–203 (1992)

[22] Moore, G.E.: A reply to my critics. In: Schilpp, P.A. (ed.) The Philosophy of G.E. Moore. The Library of Living Philosophers, vol. 4, pp. 535–677. Northwestern University, Evanston (1942)

[23] Nayak, A.C.: Iterated belief change based on epistemic entrenchment. Erkenntnis 41, 353–390 (1994)

[24] Rott, H.: Conditionals and theory change: revisions, expansions, and additions. Synthese 81, 91–113 (1989)
[25] Spohn, W.: Ordinal conditional functions: a dynamic theory of epistemic states. In: Harper, W.L., Skyrms, B. (eds.) Causation in Decision, Belief Change, and Statistics, vol. II, pp. 105–134 (1988)
[26] Veltman, F.: Defaults in update semantics. JPL 25, 221–261 (1996)

Appendix: Proofs

Proof of Lemma 1: We change the question $\mathcal{A}$ to a different question $\mathcal{A}'$, consisting of all the disjunctions of equi-plausible answers from $\mathcal{A}$. We change the answer, taking the order induced on $\mathcal{A}'$: $\bigvee_{i\in I}\mathbf{A}^i <' \bigvee_{j\in J}\mathbf{A}^j$ iff $\mathbf{A}^i < \mathbf{A}^j$ for some $i \in I, j \in J$.

Proof of Prop 2: Given two standard upgrades $\alpha = (\mathbf{A}^1, \ldots, \mathbf{A}^n)$ and $\alpha' = (\mathbf{B}^1, \ldots, \mathbf{B}^m)$, the composition $\alpha;\alpha'$ is the upgrade having $\{\mathbf{A}^i \wedge [\mathbf{A}^i]\mathbf{B}^j : i = 1, n, j = 1, m\}$ as set of answers, endowed with the anti-lexicographic order: $(\mathbf{A}^i \wedge [\mathbf{A}^i]\mathbf{B}^j) < (\mathbf{A}^k \wedge [\mathbf{A}^k]\mathbf{B}^l)$ iff either $j < l$, or $j = l$ and $i < k$.

Proof of Lemma 3: One direction is obvious: if the model stabilizes, then so do conditional beliefs. The other direction: suppose conditional beliefs stabilize at stage n. Then we have $S_n = S_m$ (else, if $t \in S_n \setminus S_m$ then $S_m \models B^{\{t\}}\bot$ but $S_n \not\models B^{\{t\}}\bot$, contradicting the stabilization of conditional beliefs at n). For $s, t \in S_n = S_m$, we have the equivalencies: $s < t$ in $\mathbf{S}_n$ iff $\mathbf{S}_n \models B^{\{s,t\}}\neg\{t\}$ iff $\mathbf{S}_m \models B^{\{s,t\}}\neg\{t\}$ iff $s < t$ in $\mathbf{S}_m$.

Proof of Lemma 4: The conclusion follows from the observation that knowledge and simple belief are definable in terms of conditional beliefs: $K\mathbf{P} = B^{\neg\mathbf{P}}\mathbf{P}$, $B\mathbf{P} = B^{\top}\mathbf{P}$.

Proof of Prop. 5: $\alpha(\mathbf{S}) \subseteq S$ for every upgrade $(\mathcal{A}, \leq)$ and every model $\mathbf{S}$. So $S_0 \supseteq S_1 \supseteq \cdots S_n \supseteq \cdots$ is an *infinite descending chain of finite sets*, hence it *must stabilize*: there exists n such that $S_n = S_m$ for all $m > n$.

Proof of Cor. 6: An update *changes only the set of possible states*. By Prop. 5, the set of states stabilizes at some finite stage, so the model itself stabilizes at that stage.

To prove Theorem 7, we first fix the setting and prove some preliminary results.

Let $\boldsymbol{\alpha} = (\alpha_n)_{n\in N}$ be an upgrade stream in standard form $\alpha_n = (\mathbf{A}_n^1, \ldots, \mathbf{A}_n^{m_n})$, and let $\mathbf{A}(n) := \mathbf{A}_n^1$ be *the "best" (most plausible) answer of* α_n. Let $\mathbf{S} = (S, \leq, \|\cdot\|, s_0)$ be a model, and let $(\mathbf{S}_n)_{n\in N}$ be the sequence of pointed models $\mathbf{S}_n = (S_n, \leq_n, \|\cdot\|, s_0)$ defined by applying in turn each of the upgrades in $\boldsymbol{\alpha}$:

$$\mathbf{S}_0 = \mathbf{S}, \text{ and } \mathbf{S}_{n+1} = \alpha_n(\mathbf{S}_n).$$

In the following we assume $\boldsymbol{\alpha}$ is *correct* with respect to $\mathbf{S}$, i.e. $s_0 \in \mathbf{A}(n)_{\mathbf{S}_n}$ for all $n \in N$.

Lemma 11. For all $n \in N$ we have $s <_n t$ for all $s \in \bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}$ and all $t \in S_n \setminus \left(\bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}\right)$.

Proof. This is by induction on n: for $n = 0$, it is trivially true (since $\bigcap_{i<0} \mathbf{A}(i)_{\mathbf{S}_i} = \emptyset$). For the induction step: we assume it true for n. After applying α_n to $\mathbf{S}_n$, we have (by the definition of an upgrade) that

$$s <_{n+1} w \text{ for all } s \in \mathbf{A}(n)_{\mathbf{S}_n} \text{ and all } w \in S_{n+1} \setminus \mathbf{A}(n)_{\mathbf{S}_n}$$

(since all $\mathbf{A}(n)$-worlds get "promoted") and also that

$$s <_{n+1} t \text{ for all } s \in \mathbf{A}(n)_{\mathbf{S}_n} \cap \left(\bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}\right) \text{ and all } t \in \mathbf{A}(n)_{\mathbf{S}_n} \setminus \left(\bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}\right)$$

(because of the induction assumption together with the fact that inside the $\mathbf{A}(n)_{\mathbf{S}_n}$-zone the old order $\leq_n$ is preserved by applying α_n). Putting these two together, and using the transitivity of $<_{n+1}$, we obtain the desired conclusion.

Lemma 12. For all $n \in N$, we have $best\, \mathbf{S}_n \subseteq \bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}$.

Proof. Suppose towards a contradiction that, for some n, there is some state $t \in best\, \mathbf{S}_n$ such that $t \notin \bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}$. By the "correctness" assumption, we have $s_0 \in \mathbf{A}(i)_{\mathbf{S}_i}$ for all i, so $s_0 \in \bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}$. By Lemma 11, $s_0 <_n t$, which contradicts with $t \in best\, \mathbf{S}_n$.

Lemma 13. For all $n \in N$, we have $best\, \mathbf{S}_n = Min_{\leq_n} \left(\bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i}\right)$.

Proof. This follows immediately from the previous Lemma, together with the definition of $best\, \mathbf{S}_n$ and of $Min_{\leq_n} P$.

Lemma 14. There exists a number n_0 s. t. $\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i} = \bigcap_{i<m} \mathbf{A}(i)_{\mathbf{S}_i}$ for all $m \geq n_0$.

Proof. The sequence $S = S_0 \supseteq \mathbf{A}(0)_{\mathbf{S}_0} \supseteq \mathbf{A}(0)_{\mathbf{S}_0} \cap \mathbf{A}(1)_{\mathbf{S}_1} \supseteq \cdots \bigcap_{i<n} \mathbf{A}(i)_{\mathbf{S}_i} \supseteq \cdots$ is an infinite descending sequence of finite sets, so it must stabilize at some stage n_0.

Proof of Theorem 7: By definition of upgrades (and by $\mathbf{A}(m)$ being the most plausible answer of α_m), we know that for all m, the order inside the $\mathbf{A}(m)_{\mathbf{S}_m}$-zone is left the same by α_m. Let now n_0 be as in Lemma 14. So we must have $\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i} \subseteq \mathbf{A}(m)_{\mathbf{S}_m}$, for all $m \geq n_0$. Hence, the order inside $\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i}$ *is left the same by all future upgrades* α_m, with $m \geq n_0$. As a consequence, we have:

$$Min_{\leq_{n_0}} \left(\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i}\right) = Min_{\leq_m} \left(\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i}\right) \text{ for all } m \geq n_0.$$

This, together with the previous two Lemmas, gives us that:

$$best\mathbf{S}_{n_0} = Min_{\leq_{n_0}} \left(\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i}\right) = Min_{\leq_m} \left(\bigcap_{i<n_0} \mathbf{A}(i)_{\mathbf{S}_i}\right) = Min_{\leq_m} \left(\bigcap_{i<m} \mathbf{A}(i)_{\mathbf{S}_i}\right) = best S_m$$

for all $m \geq n_0$. So the sequence of most plausible states stabilizes at n_0.

Proof of Corollary 8: Suppose we have a repeated upgrade $\alpha, \alpha, \ldots$, where $\alpha = [\varphi^1, \ldots, \varphi^k]$ is such that all sentences φ^i belong to doxastic-epistemic logic. Putting together Proposition 5 and Theorem 7, we know that both the knowledge (the set of all states) and the beliefs (the set of most plausible states) stabilize: there exists some n_0 such that $S_{n_0} = S_m$ and $best\,\mathbf{S}_{n_0} = best\,\mathbf{S}_m$ for all $m > n_0$. We also know that the valuation is stable. Using these (together with the full introspection of knowledge and beliefs), we can check (by induction on φ) that, for every (single-agent) doxastic-epistemic sentence φ, we have $\|\varphi\|_{\mathbf{S}_{n_0}} = \|\varphi\|_{\mathbf{S}_m}$ for all $m > n_0$. (Here, $\|\varphi\|_{\mathbf{S}}$ is the interpretation of φ in model $\mathbf{S}$, i.e. the set $\{s \in S_n : s \models_{\mathbf{S}_n} \varphi\}$.) So the interpretations of all the answers φ^i stabilize at stage n_0. Hence (by the definition of the model transformation induced by α), applying α after this stage will not produce any changes.

Proofs of Corollaries 9 and 10: These follow immediately from Theorem 7 and Corollary 8, together with the fact that a radical upgrade is correct iff it is truthful.

First-Order Linear-Time Epistemic Logic with Group Knowledge: An Axiomatisation of the Monodic Fragment

Francesco Belardinelli and Alessio Lomuscio

Department of Computing
Imperial College London, UK
{F.Belardinelli,A.Lomuscio}@imperial.ac.uk

Abstract. We investigate quantified interpreted systems, a computationally grounded semantics for a first-order temporal epistemic logic on linear time. We report a completeness result for the monodic fragment of a language that includes LTL modalities as well as distributed and common knowledge. We exemplify possible uses of the formalism by analysing message passing systems, a typical framework for distributed systems, in a first-order setting.

1 Introduction

Propositional modal logics to reason about knowledge and time have been thoroughly investigated by researchers in artificial intelligence both as regards their theoretical properties (completeness, decidability, complexity) [7,9], as well as for the specification and verification of multi-agent systems [4,28].

These temporal epistemic logics have been explored in several directions. In one line of research, epistemic modalities have been added to represent group knowledge such as distributed and common knowledge [8,10]. In another one, the temporal fragment has been modified according to different models of time (e.g., linear or branching, discrete or continuous) [16,18]. In yet another line, temporal epistemic logic has been studied at the first order [1,15].

In this paper we extend a combination of epistemic and temporal logic to the predicate level. We provide this language with a computationally grounded semantics [27] given in terms of *quantified interpreted systems* [1,2], and we present a complete axiomatisation of the *monodic* fragment of this logic, where at most one free variable appears in the scope of any modal operator. Finally, we apply this formalism to the modeling of message passing systems, a typical framework in distributed systems [17,4].

Our starting point is a number of results by Hodkinson, Wolter, and Zakharyaschev, among others, regarding the axiomatisability [22,25], decidability [15,24], and complexity [12,13] of first-order modal logics, including both positive [11,21] and negative results [14,23,26]. Specifically, we prove completeness for our first-order temporal epistemic logic via *quasimodels*. These structures

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 140–154, 2009.

have been used in [15] to prove decidability for *monodic* fragments of first-order temporal logic (FoTL) on a variety of flows of time. These investigations were further pursued in [14], where branching flows of time are analyzed, and in [11], which deals with the packed fragment of FoTL. In [12,13] the complexity of the decision problem for a number of monodic fragments of FoTL is considered.

As regards general first-order modal logic, the decidability of monodic fragments has been investigated in [24]. In [26] it is proved that first-order epistemic logic with common knowledge is not axiomatisable. However, in [23] it is shown that its monodic fragment is. Finally, this paper relies on results in [22,25]. In [25] the authors present a complete axiomatisation for the monodic fragment of FoTL on the naturals. In [22] we have a similar result for a variety of first-order epistemic logics with common knowledge. Note that none of these articles uses interpreted systems [4,19] as the underlying semantics, as we do here.

Our motivation for this contribution comes from an interest in first-order temporal epistemic formalisms to model high-level properties of multi-agent systems (MAS). While temporal epistemic logics are well understood at the propositional level, their usefulness has been demonstrated in a number of applications, and model checking tools have been developed for them [6,20], still we believe there is a growing need in web-services, security, as well as other areas, to extend these languages to the first order. As preliminary contributions to this project, in [2] we introduced a "static" version of quantified interpreted systems to model a first-order epistemic formalism. This was then extended to the temporal dimension in [1]. Differently from these previous works, here we explicitly assume linear-time operators and the natural numbers as the flow of time. Both features are crucial for applications, but they also increase the complexity of the formalism.

Scheme of the paper. In Section 2 we introduce the first-order temporal epistemic language $\mathcal{L}_m$, for a set $A = \{1, \ldots, m\}$ of agents, and in Section 3 we provide it with a semantics in terms of quantified interpreted systems and present its monodic fragment. In Section 4 we explore its expressive power in specifying message passing systems. In Sections 5 and 6 we introduce an axiomatisation for the monodic fragment of $\mathcal{L}_m$ and prove its completeness. We present detailed proofs in Appendix A.

2 Syntax

The first-order temporal epistemic language $\mathcal{L}_m$ contains individual variables $x_1, x_2, \ldots$, individual constants $c_1, c_2, \ldots$, and n-ary predicative letters $P^n{}_1, P^n{}_2, \ldots$, for $n \in \mathbb{N}$, the propositional connectives $\neg$ and $\rightarrow$, the universal quantifier $\forall$, the temporal operators $\bigcirc$ and $\mathcal{U}$, the epistemic operators K_i, for $i \in A$, D, and C. The only terms $t_1, t_2, \ldots$ in $\mathcal{L}_m$ are individual variables and constants.

Definition 1. *Formulas in $\mathcal{L}_m$ are defined in the BN form as follows:*

$$\phi ::= P^k(t_1, \ldots, t_k) \mid \neg\psi \mid \psi \rightarrow \psi' \mid \forall x\psi \mid \bigcirc\psi \mid \psi\mathcal{U}\psi' \mid K_i\psi \mid D\psi \mid C\psi$$

The formulas $\bigcirc\phi$ and $\phi\mathcal{U}\phi'$ are read as "ϕ *holds at the next step*" and "ϕ' *will hold and* ϕ *is the case until that moment*". The formula $K_i\phi$ represents

"*agent i knows ϕ*", while formulas $D\phi$ and $C\phi$ respectively mean "*ϕ is distributed knowledge*" and "*ϕ is common knowledge*" in the group A of agents.

We define the symbols $\wedge$, $\vee$, $\leftrightarrow$, $\exists$, G (always in the future), F (some time in the future) as standard. Further, $E\phi = \bigwedge_{i\in A} K_i\phi$, and for $\triangle$ equal to E or $\bigcirc$, $\triangle^k\phi$ is defined as follows for $k \in \mathbb{N}$: $\triangle^0\phi = \phi$ and $\triangle^{k+1}\phi = \triangle\triangle^k\phi$.

By $\phi[\boldsymbol{y}]$ we mean that $\boldsymbol{y} = y_1, \ldots, y_n$ are all the free variables in ϕ; while $\phi[\boldsymbol{y}/\boldsymbol{t}]$ is the formula obtained by substituting simultaneously some, possibly all, free occurrences of $\boldsymbol{y}$ in ϕ with $\boldsymbol{t} = t_1, \ldots, t_n$, renaming bounded variables.

3 Quantified Interpreted Systems

In this section we present a dynamic version of the "static" quantified interpreted systems in [2] by assuming the natural numbers $\mathbb{N}$ as the underlying flow of time. Specifically, for each agent $i \in A$ in a multi-agent system we introduce a set L_i of local states $l_i, l_i', \ldots$, and a set Act_i of actions $\alpha_i, \alpha_i', \ldots$. We consider local states and actions for the environment e as well. The set $\mathcal{S} \subseteq L_e \times L_1 \times \ldots \times L_m$ contains the global states of the MAS, while $Act \subseteq Act_e \times Act_1 \times \ldots \times Act_m$ is the set of joint actions. We also introduce a transition function $\tau : Act \to (\mathcal{S} \to \mathcal{S})$. Intuitively, $\tau(\alpha)(s) = s'$ encodes that agents can access the global state s' from s by performing the joint action α. We say that the global state s' is *reachable in one step* from s, or $s \sqsubset s'$, iff there is $\alpha \in Act$ such that $\tau(\alpha)(s) = s'$.

To represent the temporal evolution of the MAS we consider the flow of time $\langle \mathbb{N}, < \rangle$ of natural numbers $\mathbb{N}$ with the strict total order $<$. A run r over $\langle \mathcal{S}, Act, \tau, \mathbb{N} \rangle$ is a function from $\mathbb{N}$ to $\mathcal{S}$ such that $r(n) \sqsubset r(n+1)$. Intuitively, a run represents a possible evolution of the MAS according to the transition function τ and assuming $\mathbb{N}$ as the flow of time. We now define the quantified interpreted systems for the language $\mathcal{L}_m$ as follows:

Definition 2 (QIS). *A quantified interpreted system over $\langle \mathcal{S}, Act, \tau, \mathbb{N} \rangle$ is a triple $\mathcal{P} = \langle \mathcal{R}, \mathcal{D}, I \rangle$ such that (i) $\mathcal{R}$ is a non-empty set of runs over $\langle \mathcal{S}, Act, \tau, \mathbb{N} \rangle$; (ii) $\mathcal{D}$ is a non-empty set of individuals; (iii) I is an interpretation of $\mathcal{L}_m$ such that $I(c) \in \mathcal{D}$, and for $r \in \mathcal{R}$, $n \in \mathbb{N}$, $I(P^k, r, n)$ is a k-ary relation on $\mathcal{D}$.*
We denote by $\mathcal{QIS}$ the class of all quantified interpreted systems.

Following standard notation [4] a pair (r, n) is a *point* in $\mathcal{P}$. If $r(n) = \langle l_e, l_1, \ldots, l_m \rangle$ is the global state at the point (r, n), then $r_e(n) = l_e$ and $r_i(n) = l_i$ are the environment's and agent i's local state at (r, n) respectively. Further, a QIS is *synchronous* if for all $i \in A$, $r_i(n) = r_i'(n')$ implies $n = n'$, that is, time is part of the local state of any agent. $\mathcal{QIS}^{sync}$ is the class of all synchronous QIS.

Now we assign a meaning to the formulas of $\mathcal{L}_m$ in quantified interpreted systems. Let σ be an assignment from the variables to the individuals in $\mathcal{D}$, the valuation $I^\sigma(t)$ of a term t is defined as $\sigma(y)$ for $t = y$, and $I^\sigma(t) = I(c)$, for $t = c$. A variant $\sigma\binom{x}{a}$ of an assignment σ assigns $a \in \mathcal{D}$ to x and coincides with σ on all the other variables.

Definition 3. *The satisfaction relation $\models$ for $\phi \in \mathcal{L}_m$, $(r, n) \in \mathcal{P}$, and an assignment σ is defined as follows:*

$$
\begin{array}{lll}
(\mathcal{P}^\sigma, r, n) \models P^k(t_1, \ldots, t_k) & \text{iff} & \langle I^\sigma(t_1), \ldots, I^\sigma(t_k)\rangle \in I(P^k, r, n) \\
(\mathcal{P}^\sigma, r, n) \models \neg\psi & \text{iff} & (\mathcal{P}^\sigma, r, n) \not\models \psi \\
(\mathcal{P}^\sigma, r, n) \models \psi \to \psi' & \text{iff} & (\mathcal{P}^\sigma, r, n) \not\models \psi \text{ or } (\mathcal{P}^\sigma, r, n) \models \psi' \\
(\mathcal{P}^\sigma, r, n) \models \forall x\psi & \text{iff} & \text{for all } a \in \mathcal{D},\ (\mathcal{P}^{\sigma\binom{x}{a}}, r, n) \models \psi \\
(\mathcal{P}^\sigma, r, n) \models \bigcirc\psi & \text{iff} & (\mathcal{P}^\sigma, r, n+1) \models \psi \\
(\mathcal{P}^\sigma, r, n) \models \psi\mathcal{U}\psi' & \text{iff} & \text{there is } n' \geq n \text{ such that } (\mathcal{P}^\sigma, r, n') \models \psi' \\
 & & \text{and for all } n'',\ n \leq n'' < n' \text{ implies } (\mathcal{P}^\sigma, r, n'') \models \psi \\
(\mathcal{P}^\sigma, r, n) \models K_i\psi & \text{iff} & \text{for all } (r', n'), r_i(n) = r_i'(n') \text{ implies } (\mathcal{P}^\sigma, r', n') \models \psi \\
(\mathcal{P}^\sigma, r, n) \models D\psi & \text{iff} & r_i(n) = r_i'(n') \text{ for all } i \in A, \text{ implies } (\mathcal{P}^\sigma, r', n') \models \psi \\
(\mathcal{P}^\sigma, r, n) \models C\psi & \text{iff} & \text{for all } k \in \mathbb{N}, (\mathcal{P}^\sigma, r, n) \models E^k\psi
\end{array}
$$

The truth conditions for $\wedge$, $\vee$, $\leftrightarrow$, $\exists$, G, and F are defined from those above. A formula $\phi \in \mathcal{L}_m$ is *true at a point* (r, m) iff it is satisfied at (r, m) by every σ; ϕ is *valid on a QIS* $\mathcal{P}$ iff it is true at every point in $\mathcal{P}$; ϕ is *valid on a class* $\mathcal{C}$ *of QIS* iff it is valid on every QIS in $\mathcal{C}$.

3.1 The Monodic Fragment

In what follows we focus on the monodic fragment of the language $\mathcal{L}_m$.

Definition 4. *The monodic fragment* $\mathcal{L}_m^1$ *is the set of formulas* $\phi \in \mathcal{L}_m$ *such that any subformula of* ϕ *of the form* $K_i\psi$, $D\psi$, $C\psi$, $\bigcirc\psi$, *or* $\psi_1\mathcal{U}\psi_2$ *contains at most one free variable.*

The monodic fragments of a number of first-order modal logics have been thoroughly investigated [22,25,15,13,24]. In the case of $\mathcal{L}_m$ this fragment is quite expressive as it contains formulas like the following:

$$\forall y\ (\mathit{Resource}(y) \to C\ (\forall z\mathit{Available}(y, z)\mathcal{U}\exists x\mathit{Request}(x, y)) \tag{1}$$

$$D \bigcirc \forall xyz(\mathit{Request}(x, y) \to \neg\mathit{Available}(y, z)) \to$$
$$\to \bigcirc D\forall xyz(\mathit{Request}(x, y) \to \neg\mathit{Available}(y, z)) \tag{2}$$

According to (1), it is common knowledge that every resource will eventually be requested, but until that time the resource is universally available. By (2) if it is distributed knowledge that at the next step any resource is not available whenever it is requested, then at the next step it is distributed knowledge that this is the case.

Note that the monodic fragment of $\mathcal{L}_m$ contains all *de dicto* formulas, i.e., formulas where no free variable appears in the scope of modal operators, as in (2). So, the limitation is really only on *de re* formulas.

4 Message Passing Systems

In this section we model message passing systems [4,17] in the framework of QIS. A message passing system (MPS) is a MAS in which the only actions for the agents are sending and receiving messages. This setting is common to a variety of distributed systems, well beyond the realms of MAS and AI.

To define our message passing QIS we introduce a set *Msg* of messages $\mu_1, \mu_2, \ldots$, and define the local state l_i for agent i as a *history* over *Msg*, that is, a sequence of events of the form $send(i, j, \mu)$ and $rec(i, j, \mu)$, for $i, j \in A$, $\mu \in Msg$. Intuitively, $send(i, j, \mu)$ represents the event where *agent i sends to j message* μ, while the meaning of $rec(i, j, \mu)$ is that *agent i receives from j message* μ. A global state $s \in \mathcal{S}$ is a tuple $\langle l_e, l_1, \ldots, l_n \rangle$, where $l_1, \ldots, l_n$ are local states as above, and l_e contains all the events in $l_1, \ldots, l_n$.

A run r over $\langle \mathcal{S}, \mathbb{N} \rangle$ is a function from the natural numbers $\mathbb{N}$ to $\mathcal{S}$ such that:

MP1 $r_i(0)$ is a sequence of length zero, and $r_i(m+1)$ is either identical to $r_i(m)$ or results from appending an event to $r_i(m)$.

By MP1 the local states of each agent record the messages she has sent or received, and at each step at most a single event occurs to any agent. We define message passing QIS (MPQIS) as the class of quantified interpreted systems $\mathcal{P} = \langle \mathcal{R}, \mathcal{D}, I \rangle$, where $\mathcal{R}$ is a non-empty set of runs satisfying MP1, $\mathcal{D}$ contains the agents in A and the messages in *Msg*, and I is an interpretation for $\mathcal{L}_m$. We use the same notation for objects in the model and syntactic elements.

For the specification of MPS we introduce a predicative letter *Send* such that $(\mathcal{P}^\sigma, r, n) \models Send(i, j, \mu)$ iff event $send(i, j, \mu)$ occurs to agent i at time n in run r, i.e., $r_i(n)$ is the result of appending $send(i, j, \mu)$ to $r_i(n-1)$. Also, we introduce the predicate *Sent* such that $(\mathcal{P}^\sigma, r, n) \models Sent(i, j, \mu)$ iff event $send(i, j, \mu)$ occurs to agent i before time n in run r, i.e., $send(i, j, \mu)$ appears in $r_i(n)$. The predicates *Rec* and *Rec'ed* are similarly defined for event $rec(i, j, \mu)$.

Let us now explore the range of specifications that can be expressed in this formalism. A property often required in MPS is *channel reliability*. We express this by stating that every sent message is eventually received. Notice that according to the definition of message passing QIS, it is possible that a message is lost during a run of the system. We can force channel reliability by requiring the following specification on MPQIS:

$$\forall \mu (\exists ij Send(i, j, \mu) \rightarrow F \exists i' j' Rec(j', i', \mu)) \tag{3}$$

In fact, we can be more specific and require that every message is received *at most* (*at least*) k steps after being sent, or *exactly* k steps after being sent:

$$\forall \mu (\exists ij Send(i, j, \mu) \rightarrow \bigcirc^k \exists i' j' Rec'ed(j', i', \mu)) \tag{4}$$

$$\forall \mu (\exists ij Sent(i, j, \mu) \rightarrow \bigcirc^k \exists i' j' Rec(j', i', \mu)) \tag{5}$$

$$\forall \mu (\exists ij Send(i, j, \mu) \rightarrow \bigcirc^k \exists i' j' Rec(j', i', \mu)) \tag{6}$$

Note that all of (3)-(6) are monodic. In these specifications the identities of the sender and the receiver are left unspecified. So, in cases in which we are not interested in singling out the addresser and the addressee, the monodic fragment suffices.

Another property often required on MPQIS is that there are no "ghost" messages: if agent i receives a message μ, then i knows that μ must actually have been sent by some agent j. This specification is expressible as a monodic formula:

$$\forall \mu (\exists j Rec'ed(i, j, \mu) \rightarrow K_i \exists j' Sent(j', i, \mu)) \tag{7}$$

We compare (7) with a further relevant property of MPQIS, i.e., *authentication*: if agent i has received a message μ from agent j, then i knows that μ had actually been sent by j. This specification can be expressed as the *de re* version of (7):

$$\forall \mu j (Rec'ed(i, j, \mu) \rightarrow K_i Sent(j, i, \mu)) \tag{8}$$

Note that differently from (7), (8) is not monodic.

Even if we allow an agent i not to know whether a received message μ has actually been sent, that is, we reject (7), by definition of MPQIS it is distributed knowledge that a message μ has been sent and received as soon as it has been received, i.e., the following monodic formula holds:

$$\forall \mu (\exists ij Rec'ed(i, j, \mu) \rightarrow D \exists i' j' (Sent(j', i', \mu) \wedge Rec'ed(i', j', \mu)))$$

On the other hand, the corresponding formula

$$\forall \mu (\exists j Rec'ed(i, j, \mu) \rightarrow K_i \exists j' (Sent(j', i, \mu) \wedge Rec'ed(i, j', \mu)))$$

is not valid for any agent i.

Furthermore, in $\mathcal{L}^1_m$ we can express that an agent i cannot aquire the knowledge that message μ has been sent to her, other than by receiving the message:

$$\forall \mu (\exists j Sent(j, i, \mu) \rightarrow (\neg K_i \exists j' Sent(j', i, \mu) \mathcal{U} \exists j'' Rec(i, j'', \mu))$$

Finally, we might want to check whether at a certain point in the evolution of the MPQIS it will be common knowledge that a message has been sent or received:

$$\forall \mu (\exists ij Sent(i, j, \mu) \rightarrow FC(\exists i' j' Sent(i', j', \mu))) \tag{9}$$

$$\forall \mu (\exists ij Rec'ed(i, j, \mu) \rightarrow FC(\exists i' j' Rec'ed(i', j', \mu))) \tag{10}$$

From results in [4] regarding the attainability of common knowledge in systems with unreliable communication, we may infer that some assumption on channel reliability in MPQIS is needed in order to satisfy specifications (9) and (10). The conclusion we can draw from the observations above is that the monodic fragment of the language $\mathcal{L}_m$ allows for rich specifications on MPS, notwithstanding the limitation on free variables in modal contexts.

5 Axiomatisation

In this section we present a sound and complete axiomatisation of the set of monodic validities in the class of quantified interpreted systems. The system QKT^1_m is a first-order multi-modal version of the propositional epistemic system $S5$ combined with the linear temporal logic LTL.

Definition 5. *The system QKT^1_m on $\mathcal{L}^1_m$ contains the following schemes of axioms and inference rules, where $\Box$ is any of the epistemic operators K_i, for $i \in A$, D, or C, and ϕ, ψ and χ are formulas in $\mathcal{L}^1_m$:*

Taut	*classic propositional tautologies*
MP	$\phi \rightarrow \psi, \phi \Rightarrow \psi$
$K_{\bigcirc}$	$\bigcirc(\phi \rightarrow \psi) \rightarrow (\bigcirc\phi \rightarrow \bigcirc\psi)$
	$\bigcirc\neg\phi \leftrightarrow \neg \bigcirc \phi$
	$\phi \mathcal{U} \psi \leftrightarrow \psi \vee (\phi \wedge (\phi \mathcal{U} \psi))$
$Nec_{\bigcirc}$	$\phi \Rightarrow \bigcirc\phi$
	$\chi \rightarrow \neg\psi \wedge \bigcirc\chi \Rightarrow \chi \rightarrow \neg(\phi \mathcal{U} \psi)$
$K_{\Box}$	$\Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi)$
T	$\Box\phi \rightarrow \phi$
4	$\Box\phi \rightarrow \Box\Box\phi$
5	$\neg\Box\phi \rightarrow \Box\neg\Box\phi$
$Nec_{\Box}$	$\phi \Rightarrow \Box\phi$
	$K_i\phi \rightarrow D\phi$
	$C\phi \leftrightarrow (\phi \wedge EC\phi)$
	$\phi \rightarrow (\psi \wedge E\phi) \Rightarrow \phi \rightarrow C\psi$
$BF_{\bigcirc}$	$\bigcirc\forall x\phi \leftrightarrow \forall x \bigcirc \phi$
$BF_{\Box}$	$\Box\forall x\phi \leftrightarrow \forall x\Box\phi$
Ex	$\forall x\phi \rightarrow \phi[x/t]$
Gen	$\phi \rightarrow \psi[x/t] \Rightarrow \phi \rightarrow \forall x\psi$, *for* x *not free in* ϕ

The operators K_i, D and C are $S5$ modalities, while the next $\bigcirc$ and until $\mathcal{U}$ operators are axiomatised as linear-time modalities. To this we add the classic postulates *Ex* and *Gen* for quantification. We consider the standard definitions of *proof* and *theorem*: $\vdash \phi$ means that $\phi \in \mathcal{L}_m^1$ is a theorem in QKT_m^1.

It is easy to check that the axioms of QKT_m^1 are valid on every QIS and the inference rules preserve validity. As a consequence, we have the following soundness result:

Theorem 1 (Soundness). *The system* QKT_m^1 *is sound with respect to the class* $\mathcal{QIS}$ *of quantified interpreted systems.*

Thus, QKT_m^1 is sound also for the class $\mathcal{QIS}^{sync}$ of synchronous QIS.

5.1 Kripke Models

To prove the completeness of QKT_m^1 with respect to $\mathcal{QIS}$ we first introduce an appropriate class of Kripke models, and prove completeness for these models. Then we apply a correspondence result between Kripke models and QIS.

Definition 6. *A* Kripke model *for* $\mathcal{L}_m$ *is a tuple* $\mathcal{M} = \langle\langle \mathbb{N}_j, <_j\rangle_{j\in J}, \{\sim_i\}_{i\in A}, \mathcal{D}, I\rangle$ *such that (i) each* $\mathbb{N}_j$ *is a copy of the naturals with the strict total order* $<_j$*; (ii) for* $i \in A$, $\sim_i$ *is an equivalence relation on* $\bigcup_{j\in J} \mathbb{N}_j$*; (iii)* $\mathcal{D}$ *is a non-empty set of individuals; (iv) the interpretation* I *is such that* $I(c) \in \mathcal{D}$*, and for* $n_j \in \mathbb{N}_j$*,* $I(P^k, n_j)$ *is a* k*-ary relation on* $\mathcal{D}$*.*
The class of all Kripke models is denoted by $\mathcal{K}$*.*

A Kripke model is *synchronous* if for all $i \in A$, $n_j \in \mathbb{N}_j$, $n_j \sim_i n'_{j'}$ implies $n = n'$. $\mathcal{K}^{sync}$ is the class of all synchronous Kripke models. Further, let R^* be the reflexive and transitive closure of a given relation R. The satisfaction relation $\models$ for an assignment σ is inductively defined as follows:

$(\mathcal{M}^\sigma, n_j) \models P^k(t_1, \ldots, t_k)$ iff $\langle I^\sigma(t_1), \ldots, I^\sigma(t_k)\rangle \in I(P^k, n_j)$
$(\mathcal{M}^\sigma, n_j) \models \neg\psi$ iff $(\mathcal{M}^\sigma, n_j) \not\models \psi$
$(\mathcal{M}^\sigma, n_j) \models \psi \to \psi'$ iff $(\mathcal{M}^\sigma, n_j) \not\models \psi$ or $(\mathcal{M}^\sigma, n_j) \models \psi'$
$(\mathcal{M}^\sigma, n_j) \models \forall x\psi$ iff for all $a \in \mathcal{D}$, $(\mathcal{M}^{\sigma\binom{x}{a}}, n_j) \models \psi$
$(\mathcal{M}^\sigma, n_j) \models \bigcirc\psi$ iff $(\mathcal{M}^\sigma, n + 1_j) \models \psi$
$(\mathcal{M}^\sigma, n_j) \models \psi\mathcal{U}\psi'$ iff there is $n'_j \geq_j n_j$ such that $(\mathcal{M}^\sigma, n'_j) \models \psi'$ and for all n''_j, $n_j \leq_j n''_j <_j n'_j$ implies $(\mathcal{M}^\sigma, n''_j) \models \psi$
$(\mathcal{M}^\sigma, n_j) \models K_i\psi$ iff for all $n'_{j'}$, $n_j \sim_i n'_{j'}$ implies $(\mathcal{M}^\sigma, n'_{j'}) \models \psi$
$(\mathcal{M}^\sigma, n_j) \models D\psi$ iff for all $n'_{j'}$, $(n_j, n'_{j'}) \in \bigcap_{i\in A} \sim_i$ implies $(\mathcal{M}^\sigma, n'_{j'}) \models \psi$
$(\mathcal{M}^\sigma, n_j) \models C\psi$ iff for all $n'_{j'}$, $(n_j, n'_{j'}) \in (\bigcup_{i\in A} \sim_i)^*$ implies $(\mathcal{M}^\sigma, n'_{j'}) \models \psi$

We compare Kripke models and quantified interpreted systems by means of a map $g : \mathcal{K} \to \mathcal{QIS}$. Let $\mathcal{M} = \langle\langle \mathbb{N}_j, <_j\rangle_{j\in J}, \{\sim_i\}_{i\in A}, \mathcal{D}, I\rangle$ be a Kripke model. For every equivalence relation $\sim_i$, for $n_j \in \mathbb{N}_j$, let the equivalence class $[n_j]_{\sim_i} = \{n'_{j'} \mid n_j \sim_i n'_{j'}\}$ be a local state for agent i, while $\mathbb{N}_j$ is the set of local states for the environment. Then define $g(\mathcal{M})$ as the tuple $\langle\mathcal{R}, \mathcal{D}, I'\rangle$ where $\mathcal{R}$ contains the runs r_j such that $r_j(n) = \langle n_j, [n_j]_{\sim_1}, \ldots, [n_j]_{\sim_m}\rangle$, $\mathcal{D}$ is the same as in $\mathcal{M}$, and $I'(P^k, r_j, n) = I(P^k, n_j)$. The structure $g(\mathcal{M})$ is a QIS that satisfies the following result:

Lemma 1. *For every $\phi \in \mathcal{L}_m$, $n \in \mathbb{N}$,*

$$(\mathcal{M}^\sigma, n_j) \models \phi \text{ iff } (g(\mathcal{M})^\sigma, r_j, n) \models \phi$$

We omit the proof of this lemma for reasons of space. Note that if $\mathcal{M}$ is synchronous, then also $g(\mathcal{M})$ is synchronous, i.e., $g : \mathcal{K}^{sync} \to \mathcal{QIS}^{sync}$.

6 Completeness

In this section we outline the main steps in the completeness proof; we refer to the appendix for definitions and detailed proofs.

The completeness of the system QKT^1_m with respect to the class $\mathcal{QIS}$ of quantified interpreted systems is proved by means of a quasimodel construction [5]. In particular, the version of quasimodels here considered combines the purely epistemic structures in [22] and the purely temporal structures in [25]. As the first step we show that for monodic formulas satisfiability in quasimodels implies satisfiability in Kripke models.

Lemma 2. *If there is a quasimodel $\mathfrak{Q}$ for a monodic formula $\phi \in \mathcal{L}^1_m$, then ϕ is satisfiable in a Kripke model.*

Note that if the quasimodel $\mathfrak{Q}$ for ϕ is synchronous, then also the Kripke model built from $\mathfrak{Q}$ in Lemma 2 is synchronous.

Next we prove the existence of such a quasimodel for ϕ.

Lemma 3. *Suppose that $\phi \in \mathcal{L}^1_m$ is a consistent monodic formula, then there exists a (synchronous) quasimodel for ϕ.*

In the proof of Lemma 3 we make use of results in [22,25] regarding purely epistemic and temporal first-order logic. By combining Lemmas 3 and 2 we can state the main result of this paper.

Theorem 2 (Completeness). *The system QKT^1_m is complete with respect to the class $\mathcal{QIS}$ of quantified interpreted systems.*

Assume that $\nvdash \phi$, then $\neg\phi$ is consistent and by Lemmas 3 and 2 there is a Kripke model $\mathcal{M}$ satisfying $\neg\phi$. By Lemma 1 the QIS $g(\mathcal{M})$ does not validate ϕ, therefore $\mathcal{QIS} \not\models \phi$. Similarly, we can prove the following result.

Theorem 3 (Completeness). *The system QKT^1_m is complete with respect to the class $\mathcal{QIS}^{sync}$ of synchronous QIS.*

We refer to the appendix for the details of the proofs.

7 Conclusions and Further Work

In this paper we analysed a quantified version of interpreted systems, and proved completeness for the system QKT^1_m defined on the monodic fragment of the first-order language $\mathcal{L}_m$, which includes linear-time modalities and epistemic operators for group knowledge. This result makes use of previous contributions on the axiomatisation of pure first-order epistemic and temporal logic [22,25]. Further, we showed that a wide range of specifications on message passing systems can be expressed in the monodic fragment of $\mathcal{L}_m$.

Still, further work is needed in this line of research. The present paper deals with the class $\mathcal{QIS}$ of all quantified interpreted systems and the class $\mathcal{QIS}^{sync}$ of synchronous QIS. In the axiomatisation QKT^1_m for these classes there is no interaction between temporal and epistemic operators, but interaction is essential to express epistemic concepts such as *perfect recall* and *no learning*. These refinements have been widely studied at the propositional level [7,9], but it is not clear to which extent these results apply to the first order. By results in [10] the set of validities in $\mathcal{L}^1_m$ on the class of QIS with perfect recall is not axiomatisable, as we have unaxiomatisability already at the propositional level. However, to our knowledge there is no result about the monodic fragment in the case that we drop either the linear-time modalities $\bigcirc$ and $\mathcal{U}$, and retain only F and G, or the common knowledge operator C. Results along this line would be of interest for the investigation of the expressive power of modal logic between propositional and full first-order.

Finally, another issue not tackled in this paper is decidability. We believe that by combining the techniques in [15,24] it is likely to find decidable monodic fragments of first-order temporal epistemic logic. However, this topic demands further investigations.

Acknowledgements. The research described in this paper was partly supported by the EC Framework 6 funded project CONTRACT (IST Project Number 034418), by the research project "Logica Modale e Conoscenza" funded by the Scuola Normale Superiore, Pisa, and by the Royal Society through an International Joint Project award to both authors.

References

1. Belardinelli, F., Lomuscio, A.: A Complete First-Order Logic of Knowledge and Time. In: Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 705–714. AAAI Press, Menlo Park (2008)
2. Belardinelli, F., Lomuscio, A.: Quantified Epistemic Logics for Reasoning about Knowledge in Multi-Agent Systems. Artificial Intelligence (to appear), http://dx.doi.org/10.1016/j.artint.2009.02.003
3. Blackburn, P., van Benthem, J., Wolter, F. (eds.): Handbook of Modal Logic. Elsevier, Amsterdam (2006)
4. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
5. Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-Dimensional Modal Logics. Elsevier, Amsterdam (2003)
6. Gammie, P., van der Meyden, R.: MCK: Model Checking the Logic of Knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)
7. Halpern, J., van der Meyden, R., Vardi, M.: Complete axiomatisations for reasoning about knowledge and time. SIAM Journal on Computing 33(3), 674–703 (2003)
8. Halpern, J., Moses, Y.: Knowledge and common knowledge in a distributed environment. Journal of the ACM 37(3), 549–587 (1990)
9. Halpern, J., Moses, Y.: A guide to completeness and complexity for modal logics of knowledge and belief. Artificial Intelligence 54, 319–379 (1992)
10. Halpern, J., Vardi, M.: The complexity of reasoning about knowledge and time 1: lower bounds. Journal of Computer and System Sciences 38(1), 195–237 (1989)
11. Hodkinson, I.: Monodic packed fragment with equality is decidable. Studia Logica 72, 185–197 (2002)
12. Hodkinson, I.: Complexity of monodic guarded fragments over linear and real time. Annals of Pure and Applied Logic 138, 94–125 (2006)
13. Hodkinson, I., Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: On the computational complexity of decidable fragments of first-order linear temporal logics. In: Proceedings of the International Symposium on Temporal Representation and Reasoning (TIME 2003), pp. 91–98. IEEE Press, Los Alamitos (2003)
14. Hodkinson, I., Wolter, F., Zakharyaschev, M.: Decidable and undecidable fragments of first-order branching temporal logics. In: Logic in Computer science (LICS 2002), pp. 393–402. IEEE Computer Society Press, Los Alamitos (2002)
15. Hodkinson, I., Wolter, F., Zakharyaschev, M.: Decidable fragment of first-order temporal logics. Annals of Pure and Applied Logic 106(1-3), 85–134 (2000)
16. van der Hoek, W., Meyer, J.-J.C., Treur, J.: Formal semantics of temporal epistemic reflection. In: Fribourg, L., Turini, F. (eds.) LOPSTR 1994 and META 1994. LNCS, vol. 883, pp. 332–352. Springer, Heidelberg (1994)
17. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21(7), 558–565 (1978)
18. van der Meyden, R.: Axioms for knowledge and time in distributed systems with perfect recall. In: Logic in Computer Science (LICS 1994), pp. 448–457. IEEE, Los Alamitos (1994)
19. Meyer, J.-J.C., van der Hoek, W.: Epistemic Logic for AI and Computer Science. Cambridge University Press, Cambridge (1995)
20. Raimondi, F., Lomuscio, A.: Automatic verification of multi-agent systems by model checking via OBDDs. Journal of Applied Logic 5(2), 235–251 (2007)

21. Reynolds, M.: Axiomatising first-order temporal logic: until and since over linear time. Studia Logica 57(2/3), 279–302 (1996)
22. Sturm, H., Wolter, F., Zakharyaschev, M.: Monodic epistemic predicate logic. In: Brewka, G., Moniz Pereira, L., Ojeda-Aciego, M., de Guzmán, I.P. (eds.) JELIA 2000. LNCS, vol. 1919, pp. 329–344. Springer, Heidelberg (2000)
23. Sturm, H., Wolter, F., Zakharyaschev, M.: Common knowledge and quantification. Economic Theory 19, 157–186 (2002)
24. Wolter, F., Zakharyaschev, M.: Decidable fragments of first-order modal logics. Journal of Symbolic Logic 66(3), 1415–1438 (2001)
25. Wolter, F., Zakharyaschev, M.: Axiomatizing the monodic fragment of first-order temporal logic. Annals of Pure and Applied Logic 118(1-2), 133–145 (2002)
26. Wolter, F.: First order common knowledge logics. Studia Logica 65(2), 249–271 (2000)
27. Wooldridge, M.: Computationally grounded theories of agency. In: Proceedings of the International Conference on Multi-Agent Systems (ICMAS 2000), pp. 13–22. IEEE Press, Los Alamitos (2000)
28. Wooldridge, M.: Reasoning about Rational Agents. MIT Press, Cambridge (2000)

A Appendix

In this appendix we provide definitions for the main concepts used in the completeness proof for the system QKT^1_m, as well as the relevant partial results. Intuitively, a quasimodel for a monodic formula $\phi \in \mathcal{L}^1_m$ is a relational structure whose points are sets of sets of subformulas of ϕ. Each set of sets of subformulas describes a "possible state of affairs", and contains sets of subformulas defining the individuals in the point.

Definition 7. *Given a formula $\phi \in \mathcal{L}^1_n$, we denote by $sub\phi$ the set of subformulas of ϕ, and define $sub_C\phi$ as $sub\phi \cup \{EC\psi \mid C\psi \in sub\phi\} \cup \{K_iC\psi \mid C\psi \in sub\phi, i \in A\}$. Further, let $sub_{C\bigcirc}\phi$ be the set $sub_C\phi \cup \{\neg\psi \mid \psi \in sub_C\phi\} \cup \{\bigcirc\psi \mid \psi \in sub_C\phi\} \cup \{\bigcirc\neg\psi \mid \psi \in sub_C\phi\}$.*

Let $sub_n\phi$ be the subset of $sub_{C\bigcirc}\phi$ containing formulas with at most n free variables, and let x be a variable not occurring in ϕ, we define $sub_x\phi$ as $\{\psi[y/x] \mid \psi[y] \in sub_1\phi\}$. Clearly, x is the only free variable in $sub_x\phi$. By $con\phi$ we denote the set of all constants occurring in ϕ.

Definition 8 (Type). *A* type *for ϕ is any subset $\mathfrak{t}$ of $sub_x\phi$ such that for every $\psi, \chi \in sub_x\phi$, (i) $\psi \wedge \chi \in \mathfrak{t}$ iff $\psi \in \mathfrak{t}$ and $\chi \in \mathfrak{t}$; (ii) $\neg\psi \in \mathfrak{t}$ iff $\psi \notin \mathfrak{t}$.*

This definition of type is completely standard [5,22,25]. Two types $\mathfrak{t}$, $\mathfrak{t}'$ *agree* on $sub_0\phi$ iff $\mathfrak{t} \cap sub_0\phi = \mathfrak{t}' \cap sub_0\phi$. Given a type $\mathfrak{t}$ for ϕ and a constant $c \in con\phi$, the pair $\langle \mathfrak{t}, c\rangle$ is called an *indexed type* for ϕ.

Also the following definition of state candidate is standard.

Definition 9 (State Candidate). *Let T be a set of types for ϕ that agree on $sub_0\phi$, and T^{con} a set containing for each $c \in con\phi$ an indexed type $\langle \mathfrak{t}, c\rangle$ such that $\mathfrak{t} \in T$, then the pair $\mathfrak{C} = \langle T, T^{con}\rangle$ is a* state candidate *for ϕ.*

Given a state candidate $\mathfrak{C} = \langle T, T^{con}\rangle$ we define formula $\alpha_{\mathfrak{C}}$ as follows:

$$\alpha_{\mathfrak{C}} := \bigwedge_{\mathfrak{t}\in T} \exists x \mathfrak{t}[x] \wedge \forall x \bigvee_{\mathfrak{t}\in T} \mathfrak{t}[x] \wedge \bigwedge_{\langle \mathfrak{t},c\rangle \in T^{con}} \mathfrak{t}[x/c]$$

A state candidate $\mathfrak{C}$ is *consistent* iff the formula $\alpha_{\mathfrak{C}}$ is consistent with QKT^1_m; consistent state candidates will be the points of our quasimodel. We now define a relation of *suitability* for types and state candidates which constitute the relational part of our quasimodel.

Definition 10. *1. A pair* $(\mathfrak{t}_1, \mathfrak{t}_2)$ *of types is* $\bigcirc$-suitable *iff the formula* $\mathfrak{t}_1 \wedge \bigcirc \mathfrak{t}_2$ *is consistent. It is* i-suitable *iff the formula* $\mathfrak{t}_1 \wedge \neg K_i \neg \mathfrak{t}_2$ *is consistent, and it is* D-suitable *iff the formula* $\mathfrak{t}_1 \wedge \neg D \neg \mathfrak{t}_2$ *is consistent*

2. A pair of state candidates $(\mathfrak{C}_1, \mathfrak{C}_2)$ *is* $\bigcirc$-suitable *iff the formula* $\alpha_{\mathfrak{C}_1} \wedge \bigcirc \alpha_{\mathfrak{C}_2}$ *is consistent. It is* i-suitable *iff the formula* $\alpha_{\mathfrak{C}_1} \wedge \neg K_i \neg \alpha_{\mathfrak{C}_2}$ *is consistent, and it is* D-suitable *iff the formula* $\alpha_{\mathfrak{C}_1} \wedge \neg D \neg \alpha_{\mathfrak{C}_2}$ *is consistent.*

We now introduce the frame underlying the quasimodel for ϕ.

Definition 11 (Frame). *Let* $A^+ = A \cup \{D\}$. *A* frame $\mathcal{F}$ *is a tuple* $\langle \langle \mathbb{N}_j, <_j \rangle_{j\in J}, \{\prec_l\}_{l\in A^+} \rangle$ *such that (i) each* $\mathbb{N}_j$ *is a copy of the natural numbers with the strict total order* $<_j$*; (ii) the pair* $\langle \bigcup_{j\in J} \mathbb{N}_j, \bigcup_{l\in A^+} \prec_l \rangle$ *is a set of disjoint intransitive trees*[1].

A frame is *synchronous* if for all $l \in A^+$, $n = n'$ whenever $n_j \prec_l n'_{j'}$. Further, we introduce state functions mapping points in $\mathcal{F}$ to state candidates.

Definition 12 (State Function). *A* state function *for* ϕ *over* $\mathcal{F}$ *is a map* $\mathfrak{f}$ *associating with each* $n_j \in \mathcal{F}$ *a consistent state candidate* $\mathfrak{f}(n_j) = \mathfrak{C}_{n_j}$ *for* ϕ *such that (i) the domain of* $\mathfrak{f}$ *is not empty; (ii) if* $\mathfrak{f}$ *is defined on* n_j*, then* $\mathfrak{f}$ *is defined on* $n + 1_j$*; (iii) if* $\mathfrak{f}$ *is defined on* n_j *and* $n_j \prec_l n'_{j'}$ *then* $\mathfrak{f}$ *is defined on* $n'_{j'}$.

This definition of state function takes into account also the case of synchronous systems. In what follows we often do not distinguish between a state n_j and its associated state candidate $\mathfrak{f}(n_j) = \mathfrak{C}_{n_j}$.

Finally, we provide the definition of *objects*, which correspond to the *runs* in [22,25]. We choose this name to avoid confusion with the runs in QIS.

Definition 13 (Object). *Let* $\mathfrak{f}$ *be a state function for* ϕ *over* $\mathcal{F}$. *An* object *in* $\langle \mathcal{F}, \mathfrak{f} \rangle$ *is a map* ρ *associating with every* $n_j \in \mathbb{N}_j$ *a type* $\rho(n_j)$ *in* T_{n_j} *such that*

1. *the pairs* $(\rho(n_j), \rho(n + 1_j))$ *are* $\bigcirc$*-suitable;*
2. $\rho(n_j)$ *and* $\rho(n'_{j'})$ *are* l*-suitable whenever* $n_j \prec_l n'_{j'}$*;*
3. *if* $\chi \mathcal{U} \psi \in \rho(n_j)$ *then there is* $n' \geq n$ *such that* $\psi \in \rho(n'_j)$ *and* $\chi \in \rho(n''_j)$ *for all* $n \leq n'' < n'$*;*
4. *if* $\neg K_i \psi \in \rho(n_j)$ *then for some* $n'_{j'}$, $\rho(n_j) \prec_i \rho(n'_{j'})$ *and* $\psi \notin \rho(n'_{j'})$*;*

[1] The pair $\langle U, R \rangle$ is an intransitive tree iff (i) there is a root $u_0 \in U$ such that $u_0 R^* u$ for every $u \in U$; (ii) for every $u \in U$ the set $\{u \in U \mid uR^*u\}$ is finite and linearly ordered by R^*; (iii) every $u \in U$ but the root u_0 has exactly one predecessor; (iv) the root u_0 is irreflexive.

5. *if* $\neg D\psi \in \rho(n_j)$ *then for some* $n'_{j'}$, $\rho(n_j) \prec_D \rho(n'_{j'})$ *and* $\psi \notin \rho(n'_{j'})$;
6. *if* $\neg C\psi \in \rho(n_j)$ *then for some* $n'_{j'}$, $(\rho(n_j), \rho(n'_{j'})) \in (\bigcup_{l\in A^+} \prec_l)^*$ *and* $\psi \notin \rho(n'_{j'})$.

A map ρ associating with every $n_j \in \mathbb{N}_j$ a type $\rho(n_j) \in T_{n_j}$ such that only (1) and (3) hold is a *temporal* object. Similarly, a map ρ associating with every $n_j \in \mathbb{N}_j$ a type $\rho(n_j) \in T_{n_j}$ such that only (2) and (4)-(6) hold is an *epistemic* object. Now we have all the elements to give the definition of quasimodels.

Definition 14 (Quasimodel). *A* quasimodel *for* ϕ *is a tuple* $\mathfrak{Q} = \langle \mathcal{F}, \mathfrak{f}, \mathcal{O}\rangle$ *such that* $\mathfrak{f}$ *is a state function over* $\mathcal{F}$, *and*

1. $\phi \in \mathfrak{t}$, *for some* $\mathfrak{t} \in T_{n_j}$ *and* $T_{n_j} \in \mathfrak{C}_{n_j}$
2. *every pair* $(\mathfrak{C}_{n_j}, \mathfrak{C}_{n+1_j})$ *is* $\bigcirc$*-suitable, and every pair* $(\mathfrak{C}_{n_j}, \mathfrak{C}_{n'_{j'}})$ *is* l*-suitable whenever* $n_j \prec_l n'_{j'}$
3. *for every* $\mathfrak{t} \in T_{n_j}$ *there exists an object* $\rho \in \mathcal{O}$ *such that* $\rho(n_j) = \mathfrak{t}$
4. *for every* $c \in con\phi$, *the function* ρ_c *such that* $\rho_c(n_j) = \mathfrak{t}$, *for* $\langle \mathfrak{t}, c\rangle \in T^{con}_{n_j}$ *is an object in* $\mathcal{O}$.

We can now prove Lemma 2.

Lemma 2 *If there is a quasimodel* $\mathfrak{Q}$ *for a monodic formula* $\phi \in \mathcal{L}^1_m$, *then* ϕ *is satisfiable in a Kripke model.*

Proof. The proof of this lemma is similar to those for Lemmas 11.72 and 12.9 in [5].

First of all, for every monodic formula $\psi \in \mathcal{L}^1_m$ of the form $K_i\chi$, $D\chi$, $C\chi$, $\bigcirc\chi$, or $\chi_1\mathcal{U}\chi_2$ we consider a k-ary predicate P^k_ψ, for k equal to 0 or 1. The formula $P^k_\psi(x)$ is the *surrogate* of ψ. Given a formula $\phi \in \mathcal{L}^1_m$ we denote by $\overline{\phi}$ the formula obtained from ϕ by substituting all its modal subformulas which are not within the scope of another modal operator by their surrogates.

Since every state candidate $\mathfrak{C}$ in the quasimodel $\mathfrak{Q}$ is consistent and the system QKT^1_m is based on first-order logic, the formula $\overline{\alpha}_{\mathfrak{C}}$ is consistent with first-order (non-modal) logic. As a consequence, by completeness of first-order logic, there is a first-order structure $\mathcal{I} = \langle I, \mathcal{D}\rangle$, where $\mathcal{D}$ is a non-empty set of individuals and I is an interpretation on $\mathcal{D}$, which satisfies $\overline{\alpha}_{\mathfrak{C}}$, that is, $I^\sigma \models \overline{\alpha}_{\mathfrak{C}}$ for some assignment σ to $\mathcal{D}$.

Now, we consider a cardinal $\kappa \geq \aleph_0$ greater than the cardinality of the set $\mathcal{O}$ of all objects in $\mathfrak{Q}$, and define $\mathcal{D} = \{\langle \rho, \xi\rangle \mid \rho \in \mathcal{O}, \xi < \kappa\}$. By the theory of first-order logic, we can assume without loss of generality that $\mathcal{D}$ is the domain of the first-order structure $\mathcal{I}_{n_j} = \langle I_{n_j}, \mathcal{D}\rangle$ satisfying $\overline{\alpha}_{\mathfrak{C}_{n_j}}$, that is, all structures $\mathcal{I}_{n_j}$ share a common domain $\mathcal{D}$, and for every $\mathfrak{t} \in T_{n_j}$, $\langle \rho, \xi\rangle \in \mathcal{D}$, $\rho(n_j) = \mathfrak{t}$ iff $I^\sigma_{n_j} \models \overline{\mathfrak{t}}[x]$, for $\sigma(x) = \langle \rho, \xi\rangle$. Moreover, $I_{n_j}(c) = \langle \rho, 0\rangle$, for every $c \in con\phi$.

Let us now define the Kripke model $\mathcal{M}$. Let $\mathcal{F} = \langle \langle \mathbb{N}_j, <_j\rangle_{j\in J}, \{\prec_l\}_{l\in A^+}\rangle$ be the frame of the quasimodel $\mathfrak{Q}$, we define $\mathcal{M}$ as $\langle \langle \mathbb{N}_j, <_j\rangle_{j\in J}, \{R_i\}_{i\in A}, \mathcal{D}, I\rangle$ where each sequence $\mathbb{N}_j$ of naturals in $\mathcal{F}$ belongs also to $\mathcal{M}$; each relation R_i is the reflexive, symmetric and transitive closure of $\prec_i \cup \prec_D$; $\mathcal{D}$ is defined as above; and the interpretation I is obtained by gluing together the various I_{n_j}.

By induction on the length of $\psi \in sub_x\phi$ we can show that for every σ,

$$I^{\sigma}_{n_j} \models \overline{\psi} \text{ iff } (\mathcal{M}^{\sigma}, n_j) \models \psi$$

The base of induction follows by definition of I. The step for propositional connectives and quantifiers follows by the induction hypothesis and equations $\overline{\psi_1 \to \psi_2} = \overline{\psi_1} \to \overline{\psi_2}$, $\overline{\neg\psi_1} = \neg\overline{\psi_1}$, $\overline{\forall x\psi_1} = \forall x\overline{\psi_1}$. To deal with modal operators we state without proof the following remark, the relevant cases directly follow.

Remark 1. For every $\rho \in \mathcal{O}$ and $n_j \in \mathbb{N}_j$,

(*i*) $\bigcirc\psi \in \rho(n_j)$ iff $\psi \in \rho(n+1_j)$

(*ii*) $\psi\mathcal{U}\chi \in \rho(n_j)$ iff there exists $n'_j \geq_j n_j$ such that $\chi \in \rho(n'_j)$ and for every $n_j \leq_j n''_j < n'_j, \psi \in \rho(n''_j)$

(*iii*) $K_i\psi \in \rho(n_j)$ iff for every $n'_{j'}, n_j R_i n'_{j'}$ implies $\psi \in \rho(n'_{j'})$

(*iv*) $D\psi \in \rho(n_j)$ iff for every $n'_{j'}, (n_j, n'_{j'}) \in \bigcap_{i\in A} R_i$ implies $\psi \in \rho(n'_{j'})$

(*v*) $C\psi \in \rho(n_j)$ iff for every $n'_{j'}, (n_j, n'_{j'}) \in \left(\bigcup_{i\in A} R_i\right)^*$ implies $\psi \in \rho(n'_{j'})$

The proof of this remark is similar to the one for Lemma 12.10 in [5]. To complete the proof of Lemma 2 we remark that by definition of quasimodel $\phi \in \mathfrak{t}$, for some $\mathfrak{t} \in T_{n_j}$ and $T_{n_j} \in \mathfrak{C}_{n_j}$, therefore we have that ϕ is satisfied in the Kripke model $\mathcal{M}$. □

Note that if $\mathfrak{Q}$ is a synchronous quasimodel for ϕ, then the Kripke model built from $\mathfrak{Q}$ in Theorem 2 is also synchronous.

Now it is left to prove the existence of such a quasimodel for ϕ.

Lemma 3 *Suppose that $\phi \in \mathcal{L}^1_m$ is a consistent monodic formula, then there exists a (synchronous) quasimodel for ϕ.*

In the proof we use the following partial results. These lemmas, which we state without proof, are modifications of Lemmas 11.73 and 12.11 in [5].

Lemma 4. *Let $\mathfrak{C}$ be a consistent state candidate, then we can construct an infinite sequence $\{\mathfrak{C}_n\}_{n\in\mathbb{N}}$ of state candidates such that (i) every pair $(\mathfrak{C}_n, \mathfrak{C}_{n+1})$ is $\bigcirc$-suitable; (ii) for every $\mathfrak{t} \in T_n$ there exists a temporal object ρ such that $\rho(n) = \mathfrak{t}$; (iii) for $c \in con\phi$, the function ρ_c such that $\rho_c(n) = \mathfrak{t}$, for $\langle \mathfrak{t}, c\rangle \in T^{con}_n$, is a temporal object.*

Lemma 5. *Let $\mathfrak{C}$ be a consistent state candidate, then we can construct a structure $\mathcal{W} = \langle W, \prec_1, \ldots, \prec_m, \prec_D\rangle$ such that W is a non-empty set of state candidates, and the pair $\langle W, \bigcup_{l\in A^+} \prec_l\rangle$ is a tree. Further, (i) $\mathfrak{C} \prec_l \mathfrak{C}'$ only if $\mathfrak{C}$ and $\mathfrak{C}'$ are l-suitable; (ii) for every $\mathfrak{t} \in T$, $w \in W$, there exists an epistemic object ρ such that $\rho(w) = \mathfrak{t}$; (iii) for $c \in con\phi$, the function ρ_c such that $\rho_c(w) = \mathfrak{t}$, for $\langle \mathfrak{t}, c\rangle \in T^{con}_w$, is an epistemic object.*

We can now prove Lemma 3.

Proof. Let π_ϕ be the disjunction of all formulas $\alpha_{\mathfrak{C}}$, for all state candidates for ϕ. Note that $\overline{\pi}_\phi$ is true in every first-order model, so by completeness we have that $\vdash \pi_\phi$. Since ϕ is consistent, also $\phi \wedge \pi_\phi$ is consistent. Then there is a consistent state candidate $\mathfrak{C} = \langle T, T^{con}\rangle$ such that $\phi \in \mathfrak{t}$, for some $\mathfrak{t} \in T$.

We define the structure $\langle \mathcal{F}, \mathfrak{f}\rangle$ underlying the quasimodel $\mathfrak{Q}$ in steps. At step $2n{+}1$ we extend the structure with a chain $\mathbb{N}_{\mathfrak{C}'}$ of state candidates for every state candidate $\mathfrak{C}'$ introduced at step $2n$. At stage $2n{+}2$ we provide every state candidate introduced at step $2n{+}1$ with a tree of state candidates as shown in Lemma 5.

We start with the base of induction. Define $\mathcal{F}_0 = \langle\langle \mathbb{N}_j, <_j\rangle_{j\in J_0}, \{\prec_l^0\}_{l\in A^+}\rangle$, where J_0 is empty and for every $l \in A^+$, $\prec_l^0$ is also empty. The function $\mathfrak{f}_0$ is empty as well. We also consider a set U_0 which contains only the state candidate $\mathfrak{C}$ defined above, and assume $U_{-1} = \emptyset$.

At step $2n{+}1$ the frame $\mathcal{F}_{2n+1}$ is defined as the tuple $\langle\langle \mathbb{N}_j, <_j\rangle_{j\in J_{2n+1}}, \{\prec_l^{2n+1}\}_{l\in A^+}\rangle$ such that $J_{2n+1} = J_{2n} \cup \{U_{2n} \setminus U_{2n-1}\}$, and for each $l \in A^+$, $\prec_l^{2n+1} = \prec_l^{2n}$. Further, for every $u \in U_{2n} \setminus U_{2n-1}$ by Lemma 4 there exists a sequence $\{u_k\}_{k\in\mathbb{N}}$ of state candidates such that $u_0 = u$. Thus, the state function $\mathfrak{f}_{2n}$ is extended to $\mathfrak{f}_{2n+1}$ such that $\mathfrak{f}_{2n+1}(n_u) = u_n$, for $u \in U_{2n} \setminus U_{2n-1}$, and $\mathfrak{f}_{2n+1}$ is equal to $\mathfrak{f}_{2n}$ on all the other u. Finally, $U_{2n+1} = \bigcup_{j\in J_{2n+1}} \mathbb{N}_j$.

For defining $\mathcal{F}_{2n+2}$ we take $J_{2n+2} = J_{2n+1}$. Moreover, by Lemma 5 for every $u \in U_{2n+1} \setminus U_{2n}$ there is a structure $\langle W_u, \{\prec_l\}_{l\in A^+}\rangle$ such that the pair $\langle W_u, \bigcup_{l\in A^+} \prec_l\rangle$ is a tree. We define $\prec_l^{2n+2}$ as $\prec_l^{2n+1} \cup \prec_l$, for each $l \in A^+$. Finally, $\mathfrak{f}_{2n+2} = \mathfrak{f}_{2n+1}$ and $U_{2n+2} = U_{2n+1} \cup \bigcup_{u\in U_{2n+1}\setminus U_{2n}} W_u$.

Now consider the quasimodel $\mathfrak{Q} = \langle \mathcal{F}, \mathfrak{f}, \mathcal{O}\rangle$, where $\mathcal{F} = \langle\langle \mathbb{N}_j, <_j\rangle_{j\in J}, \{\prec_l\}_{l\in A^+}\rangle$ such that $J = \bigcup_{k\in\mathbb{N}} J_k$ and $\prec_l = \bigcup_{k\in\mathbb{N}} \prec_l^k$, for $l \in A^+$, $\mathfrak{f} = \bigcup_{k\in\mathbb{N}} \mathfrak{f}_k$, and $\mathcal{O}$ is the set of all objects on $\langle \mathcal{F}, \mathfrak{f}\rangle$. By Lemmas 4 and 5 and by construction of $\mathfrak{Q}$ we can show that the objects in $\mathcal{O}$ satisfy the constraints on quasimodels. Since $\phi \in \mathfrak{t}$, for some $\mathfrak{t} \in \mathfrak{C}$ and $\mathfrak{C} \in \mathfrak{Q}$, we have that $\mathfrak{Q}$ is a quasimodel for ϕ.

Furthermore, if we want to obtain a synchronous quasimodel from the construction above we modify the step $2n{+}1$, for $n \geq 1$, as follows. For every $u \in U_{2n} \setminus U_{2n-1}$ by construction there exists a structure $\langle W_{u'}, \{\prec_l\}_{l\in A^+}\rangle$, for some $u' \in U_{2n-1}$, such that $u \in W_{u'}$. Moreover, for some $j \in J_{2n}$, $m \in \mathbb{N}$, $u' = m_j$. Now, by Lemma 4 there exists a sequence $\{u_k\}_{k\in\mathbb{N}}$ of state candidates such that $u_0 = u$, but now define the state function $\mathfrak{f}_{2n+1}$ such that $\mathfrak{f}_{2n+1}((m+k)_u) = u_k$ for $k \in \mathbb{N}$, where m is as above. It it not difficult to show that by this construction the quasimodel $\mathfrak{Q}$ for ϕ is synchronous. This completes the proof of Lemma 2. □

From Lemmas 3 and 2, Theorems 2 and 3 follow by Lemma 1. This completes the completeness proof.

On-the-Fly Macros

Hubie Chen[1] and Omer Giménez[2]

[1] Dept. of Information and Communication Technologies, UPF (Barcelona, Spain)
hubie.chen@upf.edu
[2] Dept. de Llenguatges i Sistemes Informàtics, UPC (Barcelona, Spain)
omer.gimenez@upc.edu

Abstract. We present a domain-independent algorithm for planning that computes macros in a novel way. Our algorithm computes macros "on-the-fly" for a given set of states and does not require previously learned or inferred information, nor prior domain knowledge. The algorithm is used to define new domain-independent tractable classes of classical planning that are proved to include *Blocksworld-arm* and *Towers of Hanoi*.

1 Introduction

A planning instance involves deciding if an *initial state* can be transformed into a *goal state* via the application of a sequence of *actions*. Each planning instance has a set of variables associated with it; a *state* is a mapping defined on these variables that describes the relevant features of the situation being modelled. An action describes how a state can be transformed into another. For instance, in the well-known 15 puzzle, there is a 4-by-4 grid with 15 tiles labelled with the numbers 1 through 15, and the objective is to arrange the tiles in increasing order via a sequence of moves; a move consists of sliding a tile into the unoccupied position. This puzzle can be formulated as a planning problem by letting the variables represent locations of the grid, and a state would map each location to a tile or the unoccupied position; the act of sliding a tile into the unoccupied position corresponds to the application of an action.

A *domain* is a collection of related planning instances that typically model a particular application area. For instance, the set of "sliding tile" problem instances on n-by-n grids with $n \geq 2$ over all possible initial states might be taken as a domain. An overarching research goal in planning is to develop an automated planner that can robustly solve problems in a domain-independent manner.

Macros—a term we use broadly to refer to combinations of actions—have long been studied in planning [1, 2]. Many domain-dependent applications of macros have been exhibited and studied [3–5]; also, a number of domain-independent methods for learning, inferring, filtering, and applying macros have been the topic of research continuing up to the present [6–8].

In this paper, we present a domain-independent algorithm that computes macros in a novel way. Our algorithm computes macros "on-the-fly" and does not require previously learned or inferred information, nor any prior domain knowledge. This stands in contrast to previous work on macros, since our macros are generated and applied not over a domain or even over an instance, but with respect to a "current state" s and a

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 155–169, 2009.

(small) set of related states S. This ensures that the macros generated are tailored to the state set S, and no filtering due to over-generation of macros is necessary.

We exhibit the power of our algorithm by using it to define new domain-independent tractable classes of planning that strictly extend previously defined such classes [9], and can be proved to include *Blocksworld-arm* and *Towers of Hanoi*. We believe that this is notable as theoretically defined, domain-independent tractable classes have generally struggled to incorporate construction-type domains such as these two. We hence give theoretically grounded evidence of the computational value of macros in planning.

Our algorithm. Consider the following reachability problem: given an instance Π of planning and a subset S of the state set of Π, compute the ordered pairs of states $(s, t) \in S \times S$ such that the second state t is reachable from the first state s. (By *reachable*, we mean that there is a sequence of actions that transforms the first state into the second.) This problem is clearly hard in general, as deciding if one state is reachable from another captures the complexity of planning itself (which, under the usual propositional STRIPS formulation, is known to be PSPACE-complete [10]).

A natural—albeit incomplete—algorithm for solving this reachability problem is to first compute the pairs $(s, t) \in S \times S$ such that the state t is reachable from the state s by application of a single action, and then to compute the transitive closure of these pairs. This algorithm is well-known to run in polynomial time (in the number of states and the size of the instance) but will only discover pairs for which the reachability is evidenced by plans staying within the set of states S: the algorithm is efficient but incomplete.

The algorithm that we introduce is a strict generalization of this transitive closure algorithm for the described reachability problem. We now turn to a brief, high-level description of our algorithm. Our algorithm begins by computing the pairs connected by a single action, as in the just-described algorithm, but each pair is labelled with its connecting action. The algorithm then continually applies two types of transformations to the current set of pairs until a fixed point is reached. Throughout the execution of the algorithm, every pair has an associated label which is either a single action or a macro derived by combining existing labels. The first type of transformation (which is similar to the transitive closure) is to take pairs of states having the form (s_1, s_2), (s_2, s_3) and to add the pair (s_1, s_3) whose new label is the macro obtained by "concatenating" the labels of the pairs (s_1, s_2) and (s_2, s_3). If the pair (s_1, s_3) is already contained in the current set, the algorithm replaces the label of (s_1, s_3) with the new label if the new label is "more general" than the old one. The second type of transformation is to take a state $s \in S$ and a label of an existing pair (s_1, s_2) with $s \neq s_1$, and to see if the label applied to s yields a state $t \in S$; if so, the pair (s, t) is introduced, and the same replacement procedure as before is invoked if the pair (s, t) is already present.

Our algorithm, as with the transitive closure, operates in polynomial time (as proved in the paper) and is incomplete. We want to emphasize that it can, in general, identify pairs that are not identified by the transitive closure algorithm. Why is this? Certainly, some state pairs (s, t) introduced by the first type of transformation have macro labels that, if executed one action at a time, would stay within the set S, and hence are pairs that are discovered by the transitive closure algorithm. However, the second type of transformation may apply such a macro to another state s to discover pairs $(s, t) \in S \times S$ that would *not* be discovered by the transitive closure: this occurs when a

step-by-step execution of the macro, starting from s, would leave the set S before arriving to t. Indeed, these two transformations depend on and feed off of each other: the first transformation introduces increasingly powerful macros, which in turn can be used by the second to increase the set of pairs, which in turn permits the first to derive yet more powerful macros, and so forth.

We now describe a concrete result to offer the reader a feel for the power of our algorithm. Consider the *Towers of Hanoi* domain. Figure 1 shows the initial state init and the goal state goal for the case $n = 5$. Although these two states only differ in the values of three variables—namely, d_n-on, p_1-clear and p_3-clear—it is well known that $(2^n - 1)$ disk movements are required to reach the goal state from the initial state. We prove that our algorithm, given the set $S = H(\mathsf{init}, 4)$, by which we denote the set of states within Hamming distance 4 of init, will discover macros that move any subtower of discs from one peg to another; in particular, it will derive a macro for moving the whole tower from the first to the third peg, thus solving the problem.The radius 4 arises from the local transformations that our algorithm needs to discover the macros, and is completely independent of the number n of discs of the Towers of Hanoi instance. Since the set $S = H(\mathsf{init}, 4)$ is of polynomial size $O(n^4)$, our algorithm finds the exponentially long solution in polynomial time! Indeed, this is only possible due to the use of macros, as in [11]: the macro solving the problem is defined in terms of other macros, which are in turn defined in terms of other macros, and so on.

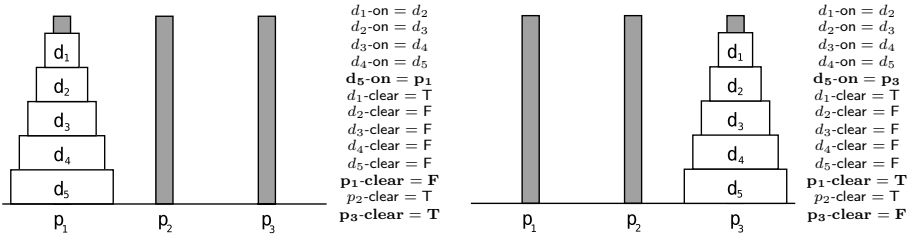


Fig. 1. Initial and goal states in the Towers of Hanoi planning problem of size $n = 5$

Our algorithm is fully domain-independent, and does not require the particular characteristics of the *Towers of Hanoi* domain to produce interesting macros. Indeed, we also obtain useful macros for the *Blocksworld-arm* domain, where a robotic arm has to move and stack blocks to reach a goal configuration. In this domain, we show that for a state s and the set $S = H(s, 4)$, our algorithm derives macros moving any subtower of blocks into the ground, preserving the subtower structure.

Towards a tractability theory of domain-independent planning. Many of the benchmark domains—such as *Gripper*, *Logistics* and *Blocksworld-arm*—can now be handled effectively and simultaneously by domain-independent planners, as borne out by empirical evidence [12]. This *empirically observed* domain-independent tractability of many common benchmark domains naturally calls for a *theoretical explanation*. By a theoretical explanation, we mean the formal definition of tractable classes of planning instances, and formal proofs that domains of interest fall into the classes. Clearly, such an

explanation could bring to the fore structural properties shared by these benchmark domains. To the best of our knowledge, research proposing tractable classes has generally had other foci, such as understanding syntactic restrictions on the set of actions [10, 13, 14], studying restrictions of the causal graph, as in [15–17, 11], or empirical evaluation of simplification rules [18]. Aligned with the present aims is the work of Hoffmann [19] that gives proofs that certain benchmark domains are solvable by local search with respect to various heuristics.

To demonstrate the efficacy of our algorithm, we use it to extend previously defined tractable classes. In particular, previous work [9] presented a complexity measure called *persistent Hamming width (PH width)*, and demonstrated that any set of instances having bounded PH width—PH width k for some constant k—is polynomial-time tractable. It was shown that both the *Gripper* and *Logistics* domains have bounded PH width, giving a uniform explanation for their tractability. In the appendix, we show that an extension of this measure yields a tractable class containing both the *Blocksworld-arm* and *Towers of Hanoi* domains, and we therefore obtain a single tractable class which captures all four of these domains. As mentioned, we believe that this is significant as theoretical treatments have generally had limited coverage of construction-type domains such as *Blocksworld-arm* and *Towers of Hanoi*.

We want to emphasize that our objective here is *not* to simply establish tractability of the domains under discussion: in them, plan generation is already well-known to be tractable on an individual, domain-dependent basis. Rather, our objective is to give a *uniform*, *domain-independent* explanation for the tractability of these domains. Neither is our goal to prove that these domains have low time complexity; again, our primary goal is to present a simple, domain-independent algorithm for which we can establish tractability of these domains with respect to the heavily-studied and mathematically robust concept of polynomial time.

Previous work on macros. Macros have long been studied in planning [1]. Early work includes [20], which developed filtering algorithms for discovered macros, and [2], which demonstrated the ability of macros to exponentially reduce the size of the search space. Some recent research on integrating macros into domain-independent planning systems is as follows. *Macro-FF* [6] is an extension of FF that has the ability to automatically learn and make use of macro-actions. *Marvin* [7] is a heuristic search planner that can form so-called macro-actions upon escaping from plateaus that can be reused for future escapes. Both of these planners participated in the International Planning Competition (IPC). A method for learning macros given an arbitrary planner and example problems from a domain is given in [8].

A more theoretical approach was taken by [11], who studied the use of macros in conjunction with causal graphs. This work gives tractability results, and in particular shows that domain-independent planners can cope with exponentially long plans in polynomial time, which is also a feature of the present work.

2 Preliminaries

An instance of the planning problem is a tuple $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ whose components are described as follows. The set V is a finite set of variables, where each variable

$v \in V$ has an associated finite domain $D(v)$. Note that variables are not necessarily propositional, that is, $D(v)$ may have any finite size. A *state* is a mapping s defined on the variables V such that $s(v) \in D(v)$ for all $v \in V$. A *partial state* is a mapping p defined on a subset $\mathsf{vars}(p)$ of the variables V such that for all $v \in \mathsf{vars}(p)$, it holds that $p(v) \in D(v)$. Then, init is a state called the initial state, and goal is a partial state. Finally, A is a set of *actions*. An action a consists of a *precondition* $\mathsf{pre}(a)$, which is a partial state, as well as a *postcondition* $\mathsf{post}(a)$, also a partial state. We sometimes denote an action a by $\langle \mathsf{pre}(a); \mathsf{post}(a) \rangle$.

The Hamming distance between two states s, s' is the number of differing variables. The set $H(s, k)$ is the set of all states s' within distance k of s. For a state or partial state s and a subset W of the variable set V, we use $s \restriction W$ to denote the partial state resulting from restricting s to W. Sometimes we will use the representation of a partial function f as the relation $\{(a, b) : f(a) = b\}$. We say that a state s is a *goal state* if $(s \restriction \mathsf{vars}(\mathsf{goal})) = \mathsf{goal}$ or, equivalently, viewing s and goal as relations, $\mathsf{goal} \subseteq s$.

An action a is *applicable* at a state s if $\mathsf{pre}(a) \subseteq s$. We define a *plan* to be a sequence of actions $P = a_1, \ldots, a_n$, with $a_i \in A$. We will always speak of actions and plans relative to some planning instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$, but we want to emphasize that when speaking of an action, the action need not be an element of A; we require only that its precondition and postcondition are partial states over Π.

Starting from a state s, we define the state resulting from s by applying a plan P, denoted by $s[P]$, inductively as follows. For the empty plan $P = \epsilon$, we define $s[\epsilon] = s$. For non-empty plans P, where $P = P', a$ and a applicable on $s[P']$, we define $s[P', a]$ as the state equal to $\mathsf{post}(a)$ on variables $v \in \mathsf{vars}(\mathsf{post}(a))$, and equal to $s[P']$ on variables $v \in V \setminus \mathsf{vars}(\mathsf{post}(a))$. If a not applicable on $s[P']$, then $s[P', a] = s[P']$.

A state s is *reachable* if there exists a plan P such that $s = \mathsf{init}[P]$. We are concerned with the problem of *plan generation*: given an instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ obtain a plan P that *solves* it, that is, a plan P such that $\mathsf{init}[P]$ is a goal state.

3 Macro Computation Algorithm

In this section, we develop our macro computation algorithm. This algorithm makes use of a number of algorithmic subroutines. Before defining them, we introduce the notion of *action graph*, the data structure on which these operations work. We emphasize that whenever we refer to actions, both in the definitions and in the algorithms, we mean precondition-postcondition pairs that need not appear in the original set of actions A.

Definition 1. *An* action graph *is a directed graph G whose vertex set, denoted by $V(G)$, is a set of states, and whose edge set, denoted by $E(G)$, consists of labelled edges that are actions, with the restriction that the label a of an edge (s, s') must be applicable at s and $s[a] = s'$. We denote the the label of an edge $e = (s_1, s_2)$ by $a = l_G(e)$ (or $l(e)$ when G is clear from context), and we say that the triplet (s, a, s') forms a* transition.

Note that for every ordered pair of vertices (s, s'), there may be at most one edge (s, s') in $E(G)$, and each edge has exactly one label.

We now give the pseudo-code of the two macro-producing operations discussed in the introduction, apply and transitive, whose aim is to add as many edges as possible to

the action graph G. They depend on the algorithmic functions better, addlabel and combine, which we define later. Notice that in our pseudocode, the assignment operator $:=$ is intended to be a value copy (as opposed to a reference copy, as in some programming languages).

Definition 2. *The pseudocode of the macro-producing operations* $\mathsf{apply}(G, A, a, s)$ *and* $\mathsf{transitive}(G, s_1, s_2, s_3)$ *is as follows. Typewise, G is an action graph, A is a set of actions, a is an action and s, s_1, s_2 and s_3 are vertices in G.*

```
apply(G, A, a, s) returns G' {
        G' := G;
        if  (a∈A ∨ a appears as a label in G') ∧
            s[a]≠s ∧ s[a]∈V(G) ∧ better(a, (s,s[a]), G)  then
                G' := addlabel(G, s, s[a], a);
        return G';
}
transitive(G, s1, s2, s3) returns G' {
        G' := G;
        if  (s1,s2)∈E(G) ∧ (s2,s3)∈E(G)  then {
                a := l(s1,s2);
                a' := l(s2,s3);
                a'' := combine(a, a');
                if  better(a'', (s1,s3), G)  then G' := addlabel(G, s1, s3, a'');
        }
        return G';
}
```

Definition 3. *The pseudocode of the functions* $\mathsf{addlabel}(G, s, s', a)$, $\mathsf{better}(a,(s, s'), G)$ *and* $\mathsf{combine}(a, a')$ *is as follows. Typewise, G is an action graph, s and s' are vertices in G, and a and a' are actions. In the particular case of* $\mathsf{combine}(a, a')$ *we require that, for some state s_1, a and a' are respectively applicable in s_1 and $s_1[a]$. This requirement is enforced in the function* transitive*, which is the only place that calls* $\mathsf{combine}(a, a')$.

```
addlabel(G, s, s', a) returns G' {
        G' := G;
        if  (s,s')∉E(G)  then place (s,s') in E(G');
        l_{G'}(s,s') := a;
        return G';
}
better(a, (s,s'), G) returns boolean  {
         if  (s,s')∉E(G)  then return TRUE;
         a' = l(s,s')
         if  pre(a)⊆pre(a') ∧ post(a)⊆post(a') ∧ a≠a'  then return TRUE;
         else return FALSE;
}
combine(a, a') returns action a'' {
        R := vars(pre(a))\vars(post(a));
        s := post(a) ∪ (pre(a)↾R);
        O := vars(post(a))\vars(post(a'));
        pr := pre(a) ∪ (pre(a')\s);
        pos := post(a') ∪ (post(a)↾O);
        return <pr; pos\pr>;
}
```

Note that in the definition of combine the partial state s represents what we know about a state if all we are told is that the action a has just been successfully executed. The following propositions identify key properties of the combine function.

Proposition 4. *Let a, a' be actions and let s be a state. The action* $\mathsf{combine}(a, a')$ *is applicable at s if and only if a is applicable at s and a' is applicable at $s[a]$. When this occurs, $s[\mathsf{combine}(a, a')]$ is equal to $s[a, a']$.*

This property ensures the *general applicability* of actions obtained from the combine procedure. That is, we may merge the pair of actions a, a' into a single action $a'' = \mathsf{combine}(a, a')$, since the sequence (a, a') and the action a'' are indistinguishable: they can be applied to the same states, and they produce the same result.

Proposition 5 (Associativity). *Assume that a_1, a_2 and a_3 are actions that are respectively applicable in states s, $s[a_1]$ and $s[a_1, a_2]$, for some s. Then, the action* $\mathsf{combine}(\mathsf{combine}(a_1, a_2), a_3)$ *is equal to the action* $\mathsf{combine}(a_1, \mathsf{combine}(a_2, a_3))$.

The following is our macro computation algorithm. As input, it takes a set of states S and a set of actions A. The running time can be bounded by $O(n|S|^3(|A| + |S|^2))$, where n denotes the number of variables.

```
compute_macros(S, A) returns G, M {
        M := {};
        V(G) := S;
        E(G) := ∅;
        do {
                A' := (A ∪ labels(E(G)));
                for all:  a∈A', s∈V(G)  {
                        G := apply(G, A, a, s);
                }
                for all:  s1,s2,s3∈V(G)  {
                        G := transitive(G, s1, s2, s3);
                        if  transitive produces a transition  then
                                M[l(s1,s3)] := l(s1,s2), l(s2,s3);
                }
        } while (some change was made to G);
        return (G, M);
}
```

The resulting action graph `G` contains the reachability information found by the algorithm. The mapping `M` contains the macros that have been used, that is, the description of how to decompose the actions appearing in G into simpler actions, up to those of A.

Understanding compute_macros. By a *combination* over A, we mean an action in A or an action that can be derived from actions in A by (possibly multiple) applications of the combine function. Clearly, all actions derived by the compute_macros algorithm are combinations of the original set of actions A. Although the actual combinations derived by the algorithm may depend on details such as the order in which the `for all` loops are executed, under certain assumptions it is possible to prove that some actions, which we call *derivable*, will be discovered by any run of the algorithm.

Definition 6. *We say that a transition (s, a, s') is* condition-minimal *with respect to a set of actions A if for any combination a' over A, if $s[a'] = s'$ then* $\mathsf{pre}(a) \subseteq \mathsf{pre}(a')$ *and* $\mathsf{post}(a) \subseteq \mathsf{post}(a')$. *In other words, either $a = a'$ or $better(a, a')$ holds true.*

Having defined the notion of a *condition-minimal* transition, we turn our attention to those that can be found by the compute_macros algorithm.

Definition 7. *An* action graph program *over a set of states S and a set of actions A is a sequence of commands $\Sigma = \sigma_1, \ldots, \sigma_n$ of the form* $\mathsf{apply}(G, A, a, s)$, *with $s \in S$, or* $\mathsf{transitive}(G, s_1, s_2, s_3)$, *with $s_1, s_2, s_3 \in S$. The execution of an action graph program takes place as follows. First, G is initialized to be the action graph with S as vertices and no edges. Then, the commands of Σ are executed in order; for each i, after σ_i is executed, G is replaced with the returned value.*

Definition 8. *An* A-condition-minimal-program *(for short, A-CM-program) over states S and actions A' is an action graph program over S and A such that when executed,* apply *is only passed pairs (a, s) such that $(s, a, s[a])$ is condition-minimal with respect to A, and the* transitive *commands produce only transitions that are condition-minimal with respect to A.*

Definition 9. *The set of (S, A)-derivable actions is the smallest set satisfying: any action of a transition produced by an A-CM-program over states S and the set of actions that are (S, A)-derivable or in A, is (S, A)-derivable.*

Lemma 10. *Relative to a planning instance Π with action set A, let s be a state. Any $(H(s, k), A)$-derivable action is discovered by a call to the function* compute_macros *with the first two arguments $H(s, k)$ and A, by which we mean that any such an action will appear as an edge label in the graph output by* compute_macros.

4 Results on compute_macros

We will present results with respect to formulations of the Blocksworld-arm and the Towers of Hanoi domains, which are based strongly on their propositional STRIPS formulations. We choose these formulations primarily to lighten the presentation, and remark that it is straightforward to verify that our proofs and results apply to the propositional formulations.

Domain 1. (Blocksworld-arm domain) We use a formulation of this domain where there is an arm. Formally, in an instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ of the Blocksworld-arm domain, there is a set of blocks B, and the variable set V is defined as $\{\mathsf{arm}\} \cup \{b\text{-on} : b \in B\} \cup \{b\text{-clear} : b \in B\}$ where $D(\mathsf{arm}) = \{\mathsf{empty}\} \cup B$ and for all $b \in B$, $D(b\text{-on}) = \{\mathsf{table}, \mathsf{arm}\} \cup B$ and $D(b\text{-clear}) = \{\mathsf{T}, \mathsf{F}\}$. The b-on variable tells what the block b is on top of, or whether it is being held by the arm, and the b-clear variable tells whether or not the block b is clear.

There are four kinds of actions.

- $\forall b \in B$, $\mathsf{pickup}_b = \langle b\text{-clear} = \mathsf{T}, b\text{-on} = \mathsf{table}, \mathsf{arm} = \mathsf{empty}; b\text{-clear} = \mathsf{F}, b\text{-on} = \mathsf{arm}, \mathsf{arm} = b\rangle$
- $\forall b \in B$, $\mathsf{putdown}_b = \langle \mathsf{arm} = b; \mathsf{arm} = \mathsf{empty}, b\text{-clear} = \mathsf{T}, b\text{-on} = \mathsf{table}\rangle$
- $\forall b, c \in B$, $\mathsf{unstack}_{b,c} = \langle b\text{-clear} = \mathsf{T}, b\text{-on} = c, \mathsf{arm} = \mathsf{empty}; b\text{-clear} = \mathsf{F}, b\text{-on} = \mathsf{arm}, \mathsf{arm} = b, c\text{-clear} = \mathsf{T}\rangle$
- $\forall b, c \in B$, $\mathsf{stack}_{b,c} = \langle \mathsf{arm} = b, c\text{-clear} = \mathsf{T}; \mathsf{arm} = \mathsf{empty}, c\text{-clear} = \mathsf{F}, b\text{-clear} = \mathsf{T}, b\text{-on} = c\rangle$

We say that a state s is *consistent* if it satisfies the following restrictions.

- $\forall b' \in B, s(b'\text{-clear}) = \mathsf{T} \Rightarrow |\{b \in B | s(b\text{-on}) = b'\}| = 0$.
- $\forall b' \in B, s(b'\text{-clear}) = \mathsf{F}, s(\mathsf{arm}) \neq b' \Rightarrow |\{b \in B | s(b\text{-on}) = b'\}| = 1$.
- $\forall b \in B, s(\mathsf{arm}) = b \Leftrightarrow s(b\text{-on}) = \mathsf{arm}, s(b\text{-clear}) = \mathsf{F}$.
- $\forall b \in B$, $\exists b_1, \ldots, b_k \in B$ such that $b\text{-on} = b_k, b_k\text{-on} = b_{k-1}, \ldots, b_1\text{-on} = \mathsf{table}$.

The planning domain Blocksworld-arm is the set of planning instances Π where init and goal are consistent, goal is total and $\mathsf{goal}(\mathsf{arm}) = \mathsf{empty}$. In any Blocksworld-arm planning instance, a state s is reachable if and only if s is consistent. In particular, all planning instances with consistent goal state are solvable. □

Definition 11. *A* pile P *of a state* s *is a non-empty sequence of blocks* $(b_1, \ldots, b_k)$ *such that* $s(b_i\text{-}\mathsf{on}) = b_{i+1}$ *for all* $i \in [1, k-1]$. *The* top *of the pile* P *is the block* $\mathsf{top}(P) = b_1$, *and the* bottom *of the pile is the block* $\mathsf{bottom}(P) = b_k$. *The* size *of* P *is* $|P| = k$. *A* sub-tower *of* s *is a pile* P *such that* $s(\mathsf{top}(P)\text{-}\mathsf{clear}) = \mathsf{T}$*; a tower is a sub-tower such that* $s(\mathsf{bottom}(P)\text{-}\mathsf{on}) = \mathsf{table}$. *The notation* $P_{\geq}(b)$ *(respectively,* $P_{>}(b)$, $P_{\leq}(b)$, $P_{<}(b)$*) denotes the sub-tower with bottom block* b *(respectively, the sub-tower stacked on* b*, and the piles supporting* b*, either including* b *or not.)*

Definition 12. *Let* $P = (b_1, \ldots, b_k)$ *be a pile,* b *and* b' *be two different blocks not in* P*, and* q *be the partial state* $\{b_1\text{-}\mathsf{clear} = \mathsf{T}, \mathsf{arm} = \mathsf{empty}, b_1\text{-}\mathsf{on} = b_2, \ldots, b_{k-1}\text{-}\mathsf{on} = b_k\}$. *We define actions with* q *as common precondition. The action* $\mathsf{subtow\text{-}table}_{P,b} = \langle q, b_k\text{-}\mathsf{on} = b; b_k\text{-}\mathsf{on} = \mathsf{table}, b\text{-}\mathsf{clear} = \mathsf{T}\rangle$ *moves a sub-tower* P *from* b *to the table. The action* $\mathsf{subtow\text{-}block}_{P,b,b'} = \langle q, b_k\text{-}\mathsf{on} = b, b'\text{-}\mathsf{clear} = \mathsf{T}; b_k\text{-}\mathsf{on} = b', b\text{-}\mathsf{clear} = \mathsf{T}, b'\text{-}\mathsf{clear} = \mathsf{F}\rangle$ *moves a sub-tower* P *from a* b *onto* b'. *The action* $\mathsf{tow\text{-}block}_{P,b'} = \langle q, b_k\text{-}\mathsf{on} = \mathsf{table}, b'\text{-}\mathsf{clear} = \mathsf{T}; b_k\text{-}\mathsf{on} = b', b'\text{-}\mathsf{clear} = \mathsf{F}\rangle$ *moves a tower* P *onto* b'.

Theorem 13. *Let* s *be a reachable state with* $s(\mathsf{arm}) = \mathsf{empty}$. *If* P *is a sub-tower of* s *and* $s(b_k\text{-}\mathsf{on}) = b$*, then* $\mathsf{subtow\text{-}table}_{P,b}$ *is* $(H(s,4), A)$*-derivable. If* P *is a sub-tower of* s*,* $s(b_k\text{-}\mathsf{on}) = b$ *and* $s(b'\text{-}\mathsf{clear}) = \mathsf{T}$*, then* $\mathsf{subtow\text{-}block}_{P,b,b'}$ *is* $(H(s,5), A)$*-derivable. If* P *is a tower of* s*,* $s(b_k\text{-}\mathsf{on}) = \mathsf{table}$ *and* $s(b'\text{-}\mathsf{clear}) = \mathsf{T}$*, then* $\mathsf{tow\text{-}block}_{P,b'}$ *is* $(H(s,4), A)$*-derivable.*

The previous theorem states that our macro computing algorithm will *always* discover these interesting actions when applied to the state s and the set $S = H(s,5)$. Note that the polynomial bound on the running time of the algorithm does not depend on the height of the subtower being moved.

Domain 2. (Towers of Hanoi domain) We study the formulation of *Towers of Hanoi* where, for every disk d, a variable stores the position (that is, the disk or the peg) the disk d is on. Formally, in an instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ of the Towers of Hanoi domain, there is a set of disks $D = \{d_1, \ldots, d_n\}$ and a set of positions $P = D \cup \{p_1, p_2, p_3\}$. We consider the partial order $<$ in the set of positions defined by $d_i < d_j$ for every $i < j$, and $d_i < p_j$ for every i and j. The set of variables V is defined as $\{d\text{-}\mathsf{on} : d \in D\} \cup \{x\text{-}\mathsf{clear} : x \in P\}$, where $D(d\text{-}\mathsf{on}) = P$ and $D(x\text{-}\mathsf{clear}) = \{\mathsf{T}, \mathsf{F}\}$.

The only actions in Towers of Hanoi are movement actions $\mathsf{move}_{d,x,x'}$ that move a disk d into a position x from a position x', provided that both d and x are clear, and $d < x$. That is, $\mathsf{move}_{d,x',x} = \langle d\text{-}\mathsf{clear} = \mathsf{T}, x\text{-}\mathsf{clear} = \mathsf{T}, d\text{-}\mathsf{on} = x'; x\text{-}\mathsf{clear} = \mathsf{F}, x'\text{-}\mathsf{clear} = \mathsf{T}, d\text{-}\mathsf{on} = x\rangle$. The planning domain is the set of planning instances Π such that the init and goal are certain predetermined total states. Namely, in both states init and goal it holds $d_i\text{-}\mathsf{on} = d_{i+1}$ for all $i \in [1, \ldots, n-1]$, $d_1\text{-}\mathsf{clear} = \mathsf{T}$, $d_i\text{-}\mathsf{clear} = \mathsf{F}$ for all $i \in [2, n]$ and $p_2\text{-}\mathsf{clear} = \mathsf{T}$. They only differ in three variables: $\mathsf{init}(d_n\text{-}\mathsf{on}) = p_1$, $\mathsf{init}(p_1\text{-}\mathsf{clear}) = \mathsf{F}$ and $\mathsf{init}(p_3\text{-}\mathsf{clear}) = \mathsf{T}$, but $\mathsf{goal}(d_n\text{-}\mathsf{on}) = p_3$, $\mathsf{goal}(p_1\text{-}\mathsf{clear}) = \mathsf{T}$ and $\mathsf{goal}(p_3\text{-}\mathsf{clear}) = \mathsf{F}$. □

Definition 14. *Let Π be a planning domain instance of Towers of Hanoi. Let i be an integer $i \in [1, k]$. Let $x = \mathsf{init}(d_i\text{-}\mathsf{on})$ and $x' \in \{p_2, p_3\}$. We define the action* $\mathsf{subtow\text{-}pos}_{i,x,x',x''} = \langle d_1\text{-}\mathsf{clear} = \mathsf{T}, d_1\text{-}\mathsf{on} = d_2, \ldots, d_{i-1}\text{-}\mathsf{on} = d_i, d_i\text{-}\mathsf{on} = x, x'\text{-}\mathsf{clear} = \mathsf{T}, x''\text{-}\mathsf{clear} = \mathsf{T}; d_i\text{-}\mathsf{on} = x'', x\text{-}\mathsf{clear} = \mathsf{T}, x''\text{-}\mathsf{clear} = \mathsf{F}\rangle$*, that is, the action that moves the tower of height i from x to x'' using the clear position x' as an intermediate position.*

Theorem 15. *Any action* $\mathsf{subtow\text{-}pos}_{i,x,x',x''}$ *is $(H(\mathsf{init}, 4), A)$-derivable.*

References

1. Fikes, R.E., Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 5(2), 189–208 (1971)
2. Korf, R.E.: Learning to solve problems by solving for macro-operators. In: Research notes in artificial intelligence. Pitman (1985)
3. Iba, G.A.: A heuristic approach to the discovery of macro-operators. Machine Learning 3(4), 285–317 (1989)
4. Junghanns, A., Schaeffer, J.: Sokoban: enhancing single-agent search using domain knowledge. Artificial Intelligence 129, 219–251 (2001)
5. Hernádvölgyi, I.: Searching for macro-operators with automatically generated heuristics. In: 14th Canadian Conference on AI, pp. 194–203 (2001)
6. Botea, A., Enzenberger, M., Müller, M., Schaeffer, J.: Macro-FF: Improving ai planning with automatically learned macro-operators. JAIR 24, 581–621 (2005)
7. Coles, A., Smith, A.: Marvin: A heuristic search planner with online macro-action learning. JAIR 28, 119–156 (2007)
8. Newton, M.A.H., Levine, J., Fox, M., Long, D.: Learning macro-actions for arbitrary planners and domains. In: ICAPS (2007)
9. Chen, H., Gimenez, O.: Act local, think global: Width notions for tractable planning. In: ICAPS (2007)
10. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence 69, 165–204 (1994)
11. Jonsson, A.: The role of macros in tractable planning over causal graphs. In: ICAPS, pp. 1936–1941 (2007)
12. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. JAIR 14, 253–302 (2001)
13. Bäckström, C., Nebel, B.: Complexity results for SAS+ planning. Computational Intelligence 11(4), 625–655 (1995)
14. Erol, K., Nau, D.S., Subrahmanian, V.S.: Complexity, decidability and undecidability results for domain-independent planning. Artificial Intelligence 76, 625–655 (1995)
15. Brafman, R., Domshlak, C.: Structure and complexity of planning with unary operators. JAIR 18, 315–349 (2003)
16. Brafman, R., Domshlak, C.: Factored planning: How, when, and when not. In: AAAI (2006)
17. Helmert, M.: The fast downward planning system. JAIR 26, 191–246 (2006)
18. Haslum, P.: Reducing accidental complexity in planning problems. In: IJCAI (2007)
19. Hoffmann, J.: Utilizing Problem Structure in Planning. LNCS (LNAI), vol. 2854. Springer, Heidelberg (2003)
20. Minton, S.: Selectively generalizing plans for problem-solving. In: IJCAI, pp. 596–599 (1985)

A Width

In this section, we present the definition of macro persistent Hamming width and present the width results on domains. For a state s, we define $\mathsf{wrong}(s)$ to be the variables that are not in the goal state, that is, $\mathsf{wrong}(s) = \{v \in \mathsf{vars}(\mathsf{goal}) \mid s(v) \neq \mathsf{goal}(v)\}$.

Definition 16. *A state s' is an* improvement *of a state s if: for all $v \in V$, if $v \in \mathsf{vars}(\mathsf{goal})$ and $s(v) = \mathsf{goal}(v)$, then $s'(v) = \mathsf{goal}(v)$; and, there exists $u \in \mathsf{vars}(\mathsf{goal})$ such that $u \in \mathsf{wrong}(s)$ and $s'(u) = \mathsf{goal}(u)$. In this case, we say that such a variable u is a variable being improved. We say that a plan P* improves *the state s if $s[P]$ is a goal state, or $s[P]$ is an improvement of s.*

We remark that in the previous definition we permit P to be the empty plan ϵ; in particular, we have that the empty plan improves any goal state.

Definition 17. *(from [9]) A planning instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ has* persistent Hamming width k *(for short,* PH width k*) if no plan exists solving Π, or for every reachable state s, there exists a plan (over A) improving s that stays within Hamming distance k of s.*

In this definition, when we say that a plan *stays within Hamming distance k* of a state s, we mean that when the plan is executed in s, all intermediate states encountered (as well as the final state) are within Hamming distance k of s.

Relative to a planning instance, we say that a state s dominates another state s' if $\{v \in V : s(v) \neq s'(v)\} \subseteq \mathsf{vars}(\mathsf{goal})$ and $\mathsf{wrong}(s) \subseteq \mathsf{wrong}(s')$; intuitively, s may differ from s' only in that it may have more variables set to their goal position.

Definition 18. *A planning instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ has* macro persistent Hamming width k *(for short,* MPH width k*) if no plan solving Π exists, or for every reachable state s dominating the initial state* init*, there exists a plan over $(H(s,k), A)$-derivable actions improving s that stays within Hamming distance k of s.*

It is straightforward to verify that if an instance has PH width k, then it has MPH width k. We now give a polynomial-time algorithm for sets of planning instances having bounded MPH width. We establish the following theorem.

Theorem 19. *Let $\mathcal{C}$ be a set of planning instances having MPH width k. The plan generation problem for $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ belonging to $\mathcal{C}$ is solvable in polynomial time via the following algorithm, in time $O(n^{3k+2}d^{3k}(|A| + (nd)^{2k}))$. Here, n denotes the number of variables $|V|$ and d denotes the maximum size of a domain.*

```
solve_mph((V, init, goal, A), k) {
        Q := empty plan;
        M := empty set of macros;
        s := init;
        while  s not a goal state  do {
                (G, M') := compute_macros(H(s,k), A);
                append M' to M;
                if  an improvement s' of s is reachable from s in G  then s := s';
                else  {print "?"; halt;}
                append l(s, s') to Q;
        }
        print M, Q;
}
```

We remark that `solve_mph` can really be viewed as an extension of an algorithm for persistent Hamming (PH) width; one essentially obtains an algorithm for PH width from `solve_mph` by replacing the call to `compute_macros` with a command that simply sets G to be the directed graph with vertex set $H(s,k)$ and an edge (s_1, s_2) present if there is an action a in A such that $s_1[a] = s_2$.

Theorem 20. *All instances of the Blocksworld-arm domain have MPH-width* 8.

According to Theorem 13, at any state s we may consider our set of applicable actions enriched by these new macro-actions. We now show how we can use them to improve any reachable state s. The proof is conceptually simple: improve s just by moving around a few piles of blocks. For instance, if $s(b\text{-on}) = b'$ but $\mathsf{goal}(b\text{-on}) = b''$, then apply action $\mathsf{subtow\text{-}table}_{P_>(b''),b''}$ (if $s(b''\text{-clear}) = \mathsf{F}$), and $\mathsf{subtow\text{-}block}_{P_{\geq}(b),b',b''}$. However, we must not forget that variables that were already in the goal state in s must remain so after the improvement. For instance, if b was on top of b' in s, then unstacking b from b' will make b'-clear change from F to T. We may try to solve this by placing something else on top of b', but then this movement may affect some other variable which was already in the goal state, and so forth.

The following lemma is a case-by-case analysis of the solution to the difficulty we have described, which allows us to prove Theorem 20.

Lemma 21. *Let Π be an instance of the Blocksworld-arm domain, and let s be a reachable state of Π such that $s(\mathsf{arm}) = \mathsf{empty}$. If a block b is such that $s(b\text{-clear}) = \mathsf{T}$ but $\mathsf{goal}(b\text{-clear}) = \mathsf{F}$, then there is a $(H(s,6), A)$-derivable action that improves the variable b-clear in s.*

With respect to Towers of Hanoi, it is clear that any instance can be solved by a single application of the $H(\mathsf{init}, 4)$-derivable action $\mathsf{subtow\text{-}pos}_{n,p_1,p_2,p_3}$. This is enough to ensure MPH-width 4, since no state dominates the initial state init other than goal.

Theorem 22. *All instances of the Towers of Hanoi domain have MPH-width* 4.

B Proofs

Proof of Lemma 10. Let $\Sigma = \sigma_1, \ldots, \sigma_n$ be an A-CM-program over $H(s,k)$ obtained by compute_macros, and let G be the resulting action graph. We prove by induction on $i \geq 1$ that after the command σ_i is executed and returns graph G_i, for every edge $(s, s') \in E(G_i)$, it holds that $(s, s') \in E(G)$ and $l_{G_i}(s, s') = l_G(s, s')$.

If σ_i is an apply command (with arguments s and a) that effects a change in the graph, then the input action must be in $l(E(G_i))$. The command σ_i can be successfully applied at G. Since G is a fixed point over all apply and transitive commands, the action a passed to apply or one that is better (according to the function better) must appear in G at $l_G(s, s[a])$. By condition-minimality of $(s, a, s[a])$, we have that $a = l_G(s, s[a])$.

If σ_i is a transitive command that produces a transition (s, a, s'), then the actions a' and a'' (from within the execution of the command), by induction hypothesis, appear in G. Since G is a fixed point over all apply and transitive commands, the action $\mathsf{combine}(a, a')$ or one that is better must appear in G at $l_G(s, s')$. By condition-minimality of $(s, \mathsf{combine}(a, a'), s')$, we have that $\mathsf{combine}(a, a') = l_G(s, s')$. □

Proof of Theorem 13. The proof has two parts. First, we show that the aforementioned actions are condition-minimal. Then, we describe how to obtain an A-CM-program that produces the actions inside $H(s,5)$. We consider the case $a = \mathsf{subtow\text{-}block}_{P,b,b'}$; the remaining actions admit similar proofs that only require Hamming distance 4.

To prove condition-minimality of a, consider a combination $C = (a_1, \ldots, a_t)$ of primitive actions from A such that $s[C] = s[a]$. We must show that the actions $\mathsf{unstack}_{b_1,b_2}, \ldots, \mathsf{unstack}_{b_k,b}, \mathsf{stack}_{b_k,b'}$ appear in C in the given relative order, and that no matter what are the remaining actions of C, this already implies that $\mathsf{pre}(a) \subseteq \mathsf{pre}(C)$ and $\mathsf{post}(a) \subseteq \mathsf{post}(C)$. We remark that the proof is not straightforward, since $\mathsf{pre}(C)$ and $\mathsf{post}(C)$ are the result of applying the combine subroutine to several actions not yet determined.

To prove that there exists an A-CM-program that produces actions subtow-table and tow-block inside $H(s,4)$ we use a mutual induction; we omit the proof here. We then use these results for subtow-block, the proof for which we sketch here. Precisely, we now show that $\mathsf{subtow\text{-}block}_{P,b,b'}$ is $(H(s,5),A)$-derivable. When $|P| = 1$, we derive $\mathsf{subtow\text{-}block}_{P,b,b'}$ by combining actions $a_1 = \mathsf{unstack}_{b_1,b}$ and $a_2 = \mathsf{stack}_{b_1,b'}$. The states $s[a_1]$ and $s[a_1,a_2]$ differ from s respectively on 4 and 3 variables, so both states lie inside $H(s,5)$. When $|P| = k$, let $P' = P_>(b_k)$ in state s. We use the derivable actions $a_1 = \mathsf{subtow\text{-}table}_{P',b_k}$, $a_2 = \mathsf{unstack}_{b_k,b}$, $a_3 = \mathsf{stack}_{b_k,b'}$ and $a_4 = \mathsf{tow\text{-}block}_{P',b_k}$. It is easy to check that the state $s[a_1,a_2,a_3]$ is the one that is furthest from s, differing at the 5 variables b-clear, b_{k-1}-on, b_k-clear, b_k-on and b'-clear. □

Proof of Theorem 15. We prove this by induction on i, the height of the subtower. The proof makes use of the classical recursive solution to Towers of Hanoi: the action $\mathsf{subtow\text{-}pos}_{i,x,x',x''}$ is the combination of actions $\mathsf{subtow\text{-}pos}_{i-1,x,x'',x'}$, $\mathsf{move}_{d_i,x,x''}$ and $\mathsf{subtow\text{-}pos}_{i-1,x',x,d_i}$.

We show that these actions are derivable from init within Hamming distance 4. This is trivial to show for the case $i = 1$, which moves a single disk; by induction, we can assume that any action of the form $\mathsf{subtow\text{-}pos}_{i-1,w,w',w''}$ is also $H(\mathsf{init},4)$-derivable. To prove that $\mathsf{subtow\text{-}pos}_{i,x,x',x''}$ is $H(\mathsf{init},4)$-derivable, we consider a state s satisfying the pre-conditions of $\mathsf{subtow\text{-}pos}_{i,x,x',x''}$ as close as possible to the initial state init. Notice that this state s is not required to be reachable (it is not even required to be consistent!) since the compute_macros algorithm takes all states within the appropiate Hamming distance into consideration. If none of the positions x, x' and x'' is a peg, let the state s be the non-consistent state obtained from init by setting d_i-on $= x$, x'-clear $= \mathsf{T}$, x''-clear $= \mathsf{T}$. We just need to check that the sequence of $H(\mathsf{init},4)$-derivable actions $\mathsf{subtow\text{-}pos}_{i-1,x,x'',x'}$, $\mathsf{move}_{d_i,x,x''}$ and $\mathsf{subtow\text{-}pos}_{i-1,x',x,d_i}$ is applicable to the state s, and that the number of differing variables after the application of any of these three actions never exceeds 4. A similar argument works for the case where some of the positions x, x' or x'' is a peg. □

Proof of Theorem 19. Let $\Pi \in \mathcal{C}$ be a planning instance such that there exists a plan for $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$. We want to show that solve_mph outputs a plan. During the execution of solve_mph, the state s can only be replaced by states that are improvements of it, and thus s always dominates the initial state init. By definition of MPH width, then, for any s encountered during execution, there exists a plan over

$(H(s,k),A)$-derivable actions improving s staying within Hamming distance k of s. By Lemma 10, all of the actions are discovered by compute_macros, and thus the reachability check in solve_mph will find an improvement.

We now perform a running time analysis of the algorithm. Let v denote the number of vertices in the graphs in compute_macros, that is, $|H(s,k)|$. We have $v \leq \binom{n}{k}d^k \in O((nd)^k)$. Let e be the maximum number of edges; we have $e = \binom{v}{2} \in O((nd)^{2k})$. The `do-while` loop in compute_macros will execute at most $2n \cdot e \in O(ne)$ times, since once an edge is introduced, its label may change at most $2n$ times, by definition of *better*. Each time this loop iterates, it uses no more than $(a+e)v+v^3$ time: *apply* can be called on no more than $(a+e)v$ inputs, and *transitive* can be called on no more than v^3 inputs. The while loop in solve_mph loops at most n times, and each time, by the previous discussion, it requires $ne((a+e)v+v^3)$ time for the call to compute_macros, and $(v+e)$ time for the reachability check. The total time is thus $O(n(ne((a+e)v+v^3)+(v+e)))$ which is $O(n^2e((a+e)v+v^3))$ which is $O(n^2e(a+e)v)$ which is $O(n^{3k+2}d^{3k}(a+(nd)^{2k}))$. □

Proof of Lemma 21. Clearly, $b = \mathsf{top}(P_1)$ for some tower P_1 of s. Let $P_2,\ldots,P_t$ be the remaining $t-1$ towers of s, and let t' be the number of towers of goal.

The proof proceeds by cases. If there is i such that $\mathsf{goal}(\mathsf{bottom}(P_i)\text{-on}) \neq \mathsf{table}$, we say we are in Case 1. Otherwise, it holds that $t \leq t'$. In particular, there are t' blocks b' such that $\mathsf{goal}(b'\text{-clear}) = \mathsf{T}$ (block b not one of them), and t blocks $b' \neq b$ such that $s(b'\text{-clear}) = \mathsf{T}$ (block b being one of them). It follows that it exists a block b' such that $\mathsf{goal}(b'\text{-clear}) = \mathsf{T}$ but $s(b'\text{-clear}) = \mathsf{F}$. We say we are in Case 2 if the block b' belongs to the tower P_1, and in Case 3 if not. Throughout this proof we say that a block b' is badly placed if $s(b'\text{-on}) \neq \mathsf{goal}(b'\text{-on})$.

Case 1. The tower P_i is wrongly placed in the table, so we are allowed to change the value of $\mathsf{bottom}(P_i)$-on without worry. If $i \neq 1$, then use $\mathsf{tow\text{-}block}_{P_i,b}$ to stack the tower P_i on top of b. If $i = 1$ and a tower P_j with $j > 1$ has a badly placed block b', then a possible solution is to insert P_1 below b'. That is, move the sub-tower $P_{\geq}(b')$ on top of P_1, and then move the new resulting tower on top of the place where b' was in state s, that is, on top of $s(b'\text{-on})$.

If $i = 1$ and no tower P_j with $j > 1$ has badly placed blocks, then consider the pile P'_i in state goal that b belongs to, and let $b' = \mathsf{top}(()P'_i)$. If block b' is in P_j for $j > 1$ in state s, then P_j would have some badly placed block, since b' and b, sharing pile P'_i in the goal state, would be in different piles in state s. So b' is in P_1, $\mathsf{goal}(b'\text{-clear}) = \mathsf{T}$ but $s(b'\text{-clear}) = \mathsf{F}$, since b is the top of P_1. It follows that the block on top of b' in pile P_1 is badly placed. To improve b-clear use actions $\mathsf{subtow\text{-}table}_{P_>(b'),b'}$ and $\mathsf{tow\text{-}block}_{P_{\leq}(b'),b}$, that is, break the tower over block b' and swap the two parts.

Note that an action like $\mathsf{tow\text{-}block}_{P_{\leq}(b'),b}$ is not derivable from s since the pile $P_{\leq}(b')$ is not a subtower of s, but it is derivable from $s' = s[\mathsf{subtow\text{-}table}_{P_>(b'),b'}]$, a state within distance 2 from s. This fact may increase the width required to discover the derivable actions; a careful examination reveals that we need up to width 5.

Case 2. Note that if Case 1 does not apply then $t \leq t'$. Let b' be the highest block in P_1 such that $s(b'\text{-clear}) = \mathsf{F}$ but $\mathsf{goal}(b'\text{-clear}) = \mathsf{T}$. If $t > 1$ and a tower P_j with $j > 1$ has a badly placed block b'', then we insert the pile $P_>(b')$ below b''. This procedure improves variables b-clear and b'-clear at the same time, but it needs width 6. If there is

a second block b'' in P_1 such that $\mathsf{goal}(b''\text{-}\mathsf{clear}) = \mathsf{T}$, then swap the sub-tower $P_>(b')$ with the pile between b' and b'', the block b'' not including. This procedure requires width 5.

If there is no second block b'' in P_1 but all the towers P_j with $j > 1$ have no badly placed blocks, it follows that either $t = 1$ or all towers P_j with $j > 1$ are exactly as in the goal state. Observe that, in this situation, the blocks of P_1 form a tower in s and in goal, but the order of the blocks in the two towers must differ: the pile $P' = P_{\leq}(b')$, which is such that $\mathsf{goal}(\mathsf{top}(P')\text{-}\mathsf{clear}) = \mathsf{T}$ and $\mathsf{goal}(\mathsf{bottom}(P')\text{-}\mathsf{on}) = \mathsf{table}$, cannot be a pile in goal. Hence there is a badly placed block below b'. This situation requires width 5.

Case 3. There is a block b' such that $s(b'\text{-}\mathsf{clear}) = \mathsf{F}$ but $\mathsf{goal}(b'\text{-}\mathsf{clear}) = \mathsf{T}$, and the block is in some tower P_i other than P_1. We just stack the sub-tower $P_>(b')$ on top of b. □

Proof of Theorem 20. Let Π be an instance of the Blocksworld-arm domain, and let s be a reachable state of Π that is not a goal state. We show how to improve one of the variables of s within Hamming distance 8. We first show how to improve the variable arm; for the remaining cases, we safely assume that $s(\mathsf{arm}) = \mathsf{goal}(\mathsf{arm}) = \mathsf{empty}$.

Improving arm. We assume $s(\mathsf{arm}) \neq \mathsf{goal}(\mathsf{arm}) = \mathsf{empty}$. We use the action $\mathsf{putdown}_b$ to place the block $b = s(\mathsf{arm})$ on the table. This improves the variable arm and may also improve b-on. Note, however, that $s(b\text{-}\mathsf{clear}) = \mathsf{F}$ (according to the given formulation, the block is not clear when the arm holds it); if $\mathsf{goal}(b\text{-}\mathsf{clear}) = \mathsf{F}$, then $\mathsf{putdown}_b$ is affecting a variable which had already the right value. In that case, we use Lemma 21 to improve b-clear. The action $\mathsf{putdown}_b$ changes the value of 3 variables, including b-clear; the action of Lemma 21 changes the value of at most 6 variables, including the common variable b-clear. Hence the combination of both is $H(s, 8)$-derivable.

Improving b-on. Assume $s(b\text{-}\mathsf{on}) = \mathsf{table}, \mathsf{goal}(b\text{-}\mathsf{on}) = b'$. If $s(b'\text{-}\mathsf{clear}) = \mathsf{F}$, then move the sub-tower $P_>(b')$ onto the table. (This changes the variable b''-on, where b'' is the block on top of b' in s, which was not in the goal state in s.) Now the block b' is clear, so we stack the tower $P_{\geq}(b)$ onto b'.

Now, assume $s(b\text{-}\mathsf{on}) = b'', \mathsf{goal}(b\text{-}\mathsf{on}) = b'$. If $s(b'\text{-}\mathsf{clear}) = \mathsf{F}$ then we can swap piles $P_>(b'')$ and $P_>(b')$, which only changes the value of two variables, neither of which was in the goal state. Otherwise, we stack $P_>(b'')$ on top of b', but then b''-clear becomes true. This is a problem if $\mathsf{goal}(b''\text{-}\mathsf{clear}) = \mathsf{F}$, so we may need to apply Lemma 21 at the current state. As in the previous case, the combination of both actions is $H(s, 8)$-derivable.

Finally, we address the case $s(b\text{-}\mathsf{on}) = b'', \mathsf{goal}(b\text{-}\mathsf{on}) = \mathsf{table}$. Move $P_{\geq}(b)$ onto the table. As in the previous case, we can apply Lemma 21 to the current state if $\mathsf{goal}(b''\text{-}\mathsf{clear}) = \mathsf{F}$. The combination is $H(s, 8)$-derivable.

Improving b-clear. Assume $s(b\text{-}\mathsf{clear}) = \mathsf{F}, \mathsf{goal}(b\text{-}\mathsf{clear}) = \mathsf{T}$. This implies that the variable b'-on, where b' is the block such that $s(b'\text{-}\mathsf{on}) = b$, is not in the goal state. Thus we can safely move the pile $P_>(b)$ onto the table to improve b-clear, which only takes width 4. To solve the case $s(b\text{-}\mathsf{clear}) = \mathsf{T}, \mathsf{goal}(b\text{-}\mathsf{clear}) = \mathsf{F}$ we just need to apply Lemma 21, which requires width 6. □

Abductive Logic Grammars

Henning Christiansen[1] and Verónica Dahl[2,3]

[1] Roskilde University, Denmark
[2] Simon Fraser University, Canada
[3] Universidad de Rovira i Virgili, Spain

Abstract. By extending logic grammars with constraint logic, we give them the ability to create knowledge bases that represent the meaning of an input string. Semantic information is thus defined through extra-grammatical means, and a sentence's meaning logically follows as a by-product of string rewriting. We formalize these ideas, and exemplify them both within and outside first-order logic, and for both fixed and dynamic knowledge bases. Within the latter variety, we consider the usual left-to-right derivations that are traditional in logic grammars, but also – in a significant departure from the norm – arbitrary (i.e., order-independent) derivations. We show that rich and accurate knowledge extraction from text can be achieved through the use of this new formalism.

1 Introduction

Natural language question-answering systems usually comprise two separate modules operating in sequence: the *analyzer*, which transform an input sentence into some meaning representation of it, and the *database* – also called the *knowledge base* if it has inferential capabilities. The meaning representation produced by the analyzer can be more or less directly mapped into a query to be processed by the knowledge base in order to produce an answer. E.g., the input sentence "Where is Waldo?" could be analyzed into the meaning representation `is_in(waldo,Place)` which would fairly directly serve as a Prolog query: `?- is_in(waldo,Place)` to be answered by a Prolog database containing such facts, e.g., {`is_in(waldo,lalaland)`, `is_in(eve,paradise)`}. Some analyzers also comprise two separate modules operating in sequence: one to give a syntactic representation of the sentence, or *parse tree*; another one to map that syntactic representation into the desired database *query*. These two phases are sometimes justified by the need to lay out the complete syntactic structure first in order to correctly interpret some sentences, e.g., to determine the effect of some natural language quantifiers over others. However it has long been recognized that syntax and semantics inform each other; for instance semantic types can be used to discard a syntactically correct sentence (such as "The sofa thinks") on the basis of semantic anomaly, and syntactic information such as whether a sentence is in active or passive voice can be used to determine, for instance, that "Adam" is the actor both in "Adam ate the apple" and in "The apple was eaten by Adam". This recognition is the basis for syntactico-semantic analyzers.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 170–181, 2009.

The semantic parts of a syntactico-semantic analyzer usually involve only linguistic concepts like number and gender, used for instance to disallow sentences such as "Witches flies"), perhaps aided by world knowledge information modules; for instance to discard "The sofa thinks", a taxonomy might be consulted which identifies "the sofa" as an inanimated object. The notion of *constraints* has been increasingly used to implement syntactico-semantic interactions, e.g., sentences of the form "X thinks" are accepted subject to the constraint that X must belong to the "human" type.

Constraint based reasoning has recently proved valuable for information systems as well, for its ability to prune the search space early, with consequent substantial gains in speed. For instance an integrity constraint that people can't live more than 120 years would block updating a database with a new person older than that, rather than allowing the new information to sneak in, only to produce some later inconsistency – or worse still, an undetected error. In general, however, information system constraints don't interact with linguistic constraints, even when they both may form part of the same overall system and might use the same kind of constraining mechanism.

In this article we propose a new family of formalisms – Abductive Logic Grammars – which uses constraint logic to implement syntactico-semantic interactions, as has been done in the past, but also to *abduce knowledge base items* as a by-product of the parse – which to the best of our knowledge has not been done before in a way fully integrated with the grammar.

2 Motivation

Classic grammars are defined through string rewriting, both in pure and in computational linguistics. This entails extra apparatus for contextual information, as we shall see in the next section. Further, the contextual information typically considered limits itself to linguistic tasks, such as checking gender and number agreement in order to avoid overgeneration, whereas the reference to what makes sense is typically not used to help disambiguation in any systematic way.

Current grammar applications, however, are expanding our concept of contextual information through their increasing need for specialized knowledge bases. E.g., a web mining system might access various domain-oriented taxonomies in order to relate "water" to subclasses such as "river" or "lake" in the process of figuring out if "clear cuts near water" refers to the forestry domain.

Knowledge representation languages supporting logic inference have for several years now been recognized as a more promising approach, e.g., web mining metadata languages have been developed to let people embed knowledge into web documents in terms of logical statements that represent it; cf. the ongoing direction of "Semantic Web". However, as argued, e.g., in [1], even this is insufficient for precise, scalable, and flexible information retrieval, since the logical statements used to represent knowledge are typically dissociated from the linguistic knowledge that natural language grammars, if used concomitantly with knowledge extraction, could contribute into the process of knowledge extraction.

We propose a seamless integration between logic statement generation and natural language analysis, by extending logic grammars with constraint logic in such a way that the process of linguistic analysis intermingles with the process of generating meaning representation, with each of these processes helping the other one out. We differ from both the classical grammar and the logic grammar traditions by combining string rewriting with "semantic processing" defined by extra-grammatical means, with first-order logic as an example. We propose, formally define, and discuss the merits and shortcomings of different variants of such extended logic grammars, in the aim of finding the best logic grammar framework that is able to create knowledge bases representing the meaning, or interpretation, of an input string, as a by-product of string rewriting.

3 Background

Classical grammar formalisms are often defined solely in terms of rewriting of strings of symbols. In two-level grammars [2], for example, contextual information is represented by additional "internal" context-free rules (independently of whether this is really the best way of representing context).

Logic grammar formalisms such as Metamorphosis Grammars [3] or DCGs [4], on the other hand, use logic terms as grammar symbols, so that string rewriting involves unification rather than simple rewriting, and that contextual information can be expressed through grammar symbol arguments. Manipulating the contextual information, however, is not easy since it must be explicitly passed along in these arguments to all concerned parts of the grammar.

The introduction into logic grammars of assumptions [5,6] and constraints [7] provided better ways to represent contextual information in logic grammars: the former by allowing to dynamically record information that becomes globally accessible over the continuation, and the latter by allowing Constraint Handling Rules [8] to suitably constrain, also in globally accessible fashion, what can be derived.

Our work leans on constraint logic (programming), with a knowledge base evolving as a "global" state during the analysis that integrates grammatical and semantic-pragmatic analysis. This is exactly the sort of processing done in a constraint solver: accommodation of new knowledge is the same as adding new constraints, the accumulated knowledge may be transformed into some normal form, and the possibility of identifying inconsistencies, equivalent to a failure-and-backtrack in a logic program. In earlier work, we have exposed this relationship [9,10,7], so here we will take this identification for granted.

4 Static Abductive Grammars with Logical Knowledge Base

We augment grammar rules with the ability to identify pieces of knowledge as they are being built and we expand our notion of a grammar into that of a grammar-with-fixed-knowledge-base, as follows.

We consider some first order logic language Σ and define a *knowledge base* as a set of Σ formulas. For simplicity of notation, we allow Σ formulas (or fragments thereof) to be represented as terms within grammar rules (which we shall define later); in the following, $\mathcal{K}$ refers to the set of such terms.

Ground terms (atoms, literals, formulas, etc.) are defined as being variable-free. An *instance* of any structure s (atoms, formulas, etc., and grammar rules to be defined) is given by a systematic replacement of variables in s. Variables are generally distinguished by initial capital letter.

A *model* M for a knowledge base K is a set of ground literals over Σ such that every formula in K evaluates to *true* in M; evaluation is defined in the usual way with the truth value of a literal not in M considered *undefined* and with $\mathit{true} \wedge \mathit{undefined} \equiv \mathit{undefined}$, $\mathit{false} \wedge \mathit{undefined} \equiv \mathit{false}$ and analogous for disjunction, negation and quantifiers. The notation $K \models \phi$ indicates that the term ϕ represents a formula which is logically entailed by knowledge base K, meaning that ϕ evaluates to *true* in any model of K; K being consistent means that there exists a model of K.

For simplicity, we explain derivation for grounded rule instances only, taking for granted that an implementation may introduce logical variables and unification whenever possible; these are standard techniques and will not be commented on here; see, e.g., [11].

Definition 1. *A* grammar-with-fixed-knowledge-base *is a 5-tuple* $\langle N, T, K, R, S\rangle$ *where*

- N *are* nonterminal symbols*, each of which has a given arity;* N' *will refer to the set of all instances of* N*, also referred to as* nonterminals*.*
- T *a set of* terminal (symbol)s*,*
- K *is the* knowledge base*, which is a satisfiable set of formulas over* Σ*,*
- R *is a set of* rules*, each of the form*
  $$N' \rightarrow (\{\mathcal{K}\} + N' + T)^*.$$
- S *is a 0-ary nonterminal called the* start symbol*.*

The elements of $N' \cup T$ *are called* grammar symbols*. Notation* ground(R) *refers to the set of instances of grammar rules in which grammar symbols are ground and in any occurrence of* $\{\phi\}$*,* ϕ *can be interpreted as a formula.*

Notice that the arrow $\rightarrow$ and the curly brackets are used as meta-symbols within grammars, similar to their usages within DCGs. We use notation $lhs \rightarrow rhs_1 \mid rhs_2 \mid \cdots$ for abbreviating grammar rules with identical left-hand side, $lhs \rightarrow rhs_1$, $lhs \rightarrow rhs_2$, etc. Be aware that the arrow symbol is also used as logical implication in Σ formulas.

Example 1. Let $\Gamma_1 = \langle N_1, T_1, K_1, R_1, s\rangle$ be a grammar-with-fixed-knowledge-base given as follows.

$$N_1 = \{s/0, np/1, vp/2\}$$
$$T_1 = \{thinks, stands, curie, marie, pierre, the, sofa, ...\}$$
$$K_1 = \{thinks(marie\_curie)\}$$

$$R_1 = \{s \to np(X)\, vp(X, Knowledge)\, \{Knowledge\},$$
$$np(marie\_curie) \to marie\ curie,$$
$$np(sofa_7) \to the\ sofa,$$
$$vp(X, thinks(X)) \to thinks,$$
$$vp(X, stands(X)) \to stands\}$$

Now we must adapt the derivation relation to include in particular the derivation of knowledge base items sanctioned by the grammar. This is done by removing from a sentential form any knowledge items that logically follow from the knowledge base, which indicated their acceptance, since only those sentential forms will go on to generate another sentential form potentially leading to a successful parse. More formally:

Definition 2. *Given a grammar-with-fixed-knowledge-base* $\Gamma = \langle N, T, K, R, S\rangle$, *the* derivation relation $\Rightarrow_\Gamma$ *is defined by the following two cases.*

$$\alpha n \beta \Rightarrow_\Gamma \alpha\gamma\beta \quad whenever \quad (n \to \gamma) \in \text{ground}(R)$$
$$\alpha\{k\}\beta \Rightarrow_\Gamma \alpha\beta \quad whenever \quad K \models k$$

Derivations and notation $\Rightarrow^*_\Gamma$ *are defined as usual. The* language $L(\Gamma)$ generated by Γ *is the set of terminal strings* $\tau \in T^*$ *for which* $S \Rightarrow^*_\Gamma \tau$. *For* $\tau \in L(\Gamma)$ *we say that* τ *is an* expression of K *and that* K *is an* interpretation *of* τ.

Example 2. We have that $s \Rightarrow^*_{\Gamma_1}$ *marie curie thinks*, which means that $K_1 = \{thinks(marie\_curie)\}$ is an interpretation of the sentence *marie curie thinks*. To see this, notice that the following rule instances are used in the derivation.

$s \to np(marie\_curie)\ vp(marie\_curie, thinks(marie\_curie))\{thinks(marie\_curie)\}$
$np(marie\_curie) \to marie\ curie$
$vp(marie\_curie, thinks(marie\_curie)) \to thinks$

We do not have $s \Rightarrow^*_{\Gamma_1}$ *the sofa stands*, as it would require the rule instance

$$s \to np(sofa_7)\ vp(sofa_7, stands(sofa_7))\ \{stands(sofa_7)\},$$

but this is not allowed in derivations of Γ_1 since $K_1 \not\models stands(sofa_7)$.

Derivations from the grammars we have considered so far can be applied for generating expressions of given knowledge and for verifying a claimed interpretation of a given sentence. This is interesting as a first proposal, but obviously inadequate for real life applications, as we should not expect at beforehand, a list given of interpretations of all possible sentences of the language, which are typically infinite in number. So we now change focus and consider interpretations as dynamic components created by the grammar for a given sentence.

Definition 3 (Unconstrained abduction problem). *An* (unconstrained) abductive grammar *is a 4-tuple* $\Gamma = \langle N, T, R, S\rangle$ *where the components are as in def. 1 above.*

The unconstrained abductive interpretation problem *for string* $\tau \in T^*$ *given* Γ *is the problem of finding a knowledge base* K *such that* $\tau \in L(\Gamma_K)$ *where* $\Gamma_K = \langle N, T, K, R, S\rangle$. *It this case,* K *is an* (abductive) interpretation *of* τ.

Example 3. Consider the unconstrained abductive interpretation problem for the string *marie curie thinks* given the unconstrained abductive grammar $\Gamma_2 = \langle N_1, T_1, R_1, s\rangle$, where the components are as in ex. 1. We have that $K_1 = \{thinks(marie\_curie)\}$ is an abductive interpretation, and so is $K_2 = \{thinks(marie\_curie), thinks(pierre\_curie)\}$. The string *the sofa thinks* has similar interpretations such as $K_3 = \{thinks(sofa_7)\}$.

The problem with definition 3 is that it specifies too much. While K_1 appears to be reasonable, K_2 contains information that is not accounted for in the sentence, and K_3 seems to conflict with the generally accepted understanding of sofas' intellectual capacities. We shall therefore fine-tune our definitions so that only certain predicates, predefined as abducible, can be added to a given background knowledge base. The background knowledge base may express integrity conditions to be preserved by the interpretation and similar semantic requisites.

Definition 4 (Constraint system). *A* constraint system for abduction *is a pair* $\langle A, K^{bg}\rangle$ *where* A *is a set of predicates (called* abducibles*) and* background knowledge base K^{bg}*, which is a satisfiable set of formulas over* Σ*.*

An admissible knowledge base *based on* $\langle A, K^{bg}\rangle$ *is a set* K^A *of ground literals whose predicates are in* A *such that* $K^{bg} \cup K^A$ *is consistent.*

Definition 5 (Constrained abduction problem). *A* constrained abductive grammar *is a pair* $\langle \Gamma, C\rangle$ *where* Γ *is an abductive grammar and* C *a constraint system for abduction,* $\Gamma = \langle N, T, R, S\rangle$ *and* $C = \langle A, K^{bg}\rangle$*.*

Given a constrained abductive grammar $\langle \Gamma, C\rangle$ *as above, the* constrained abductive recognition problem *for* $\tau \in T^*$ *is the problem of finding an admissible knowledge base* K^A *such that* $\tau \in L(\Gamma_{K^A})$ *where* $\Gamma_{K^A} = \langle N, T, K^{bg} \cup K^A, R, S\rangle$*. In this case,* K^A *is called a* constrained (abductive) interpretation *of* τ*, or for short, an* interpretation*.*

Such an interpretation K^A *is* minimal *whenever no proper subset of it is an interpretation of* τ *given* $\langle \Gamma, C\rangle$*.*

Example 4. We extend our sample abductive grammar Γ_2 with a constraint system as follows. Let $\Gamma_3 = \langle \Gamma_2, C\rangle$ where $C = \langle A, K^{bg}\rangle$ is the following constraint system.

$$
\begin{aligned}
A &= \{thinks/1, stands/1\} \\
K^{bg} &= \{thinks(X) \rightarrow human(X), \\
&\quad stands(X) \rightarrow (human(X) \vee thing(X)), \\
&\quad \neg(human(X) \wedge thing(X)), \\
&\quad human(marie\_curie),\ human(pierre\_curie),\ thing(sofa_7)\}
\end{aligned}
$$

The interpretations of *marie curie thinks*, K_1 and K_2 of example 3, are still valid in the constrained Γ_3, and with K_1 as the only minimal one.

However, we notice that no interpretation exists of *the sofa thinks* in Γ_3 as the information $thinks(sofa_7)$ is inconsistent with the background knowledge K_3^{bg}: having $thinks(sofa_7) \in K_A$ indicates that any model of $K_3^{bg} \cup K_A$ contains $human(sofa_7)$; together with the known fact $thing(sofa_7) \in K_3^{bg}$, this produces falsity of $\neg(human(X) \wedge thing(X))$.

Example 5. We extend our sample abductive grammar with additional rules and change the knowledge base so it fits discourse analysis; the new nonterminal d represents discourses alias sequences of sentences.

We consider here the task of identifying the classes to which entities appearing as *np*s belong, i.e., we forget about *marie_curie* being human, etc., and try to learn that fact from what is expressed directly in the discourse. Let $\Gamma_4 = \langle\langle N_4, T_4, R_4, d\rangle, C_4\rangle$ be the constrained abductive grammar given as follows.

$$\begin{aligned}
N_4 &= N_1 \cup \{d/0, s/0\} \\
T_4 &= T_1 \\
R_4 &= R_1 \cup \{d \to s \mid s\ d\} \\
C_4 &= \langle A_4, K_4^{bg}\rangle \\
A_4 &= \{human/1, thing/1\} \\
K_4^{bg} &= \{thinks(X) \leftrightarrow human(X), \\
&\qquad stands(X) \leftrightarrow (human(X) \vee thing(X)), \\
&\qquad \neg(human(X) \wedge thing(X))\}
\end{aligned}$$

The only minimal interpretation of *marie curie thinks* is now $\{human(marie\_curie)\}$. The sentence *marie curie stands* has two minimal explanations, $\{human(marie\_curie)\}$ and $\{thing(marie\_curie)\}$; the discourse combining the two sentences has exactly $\{human(marie\_curie)\}$ as its minimal interpretation.

The previous examples indicate some general problems that abductive logic programming avoids by restricting knowledge bases to logic programs in which no abducibles can appear in the head of a rule. We will consider how the possible minimal interpretations of *marie curie stands* vary with the choice of abducible predicates A, when we assume the background formula $stands(X) \leftrightarrow (human(X) \vee thing(X))$. With $A = \{human/1, thing/1\}$, we get two minimal interpretations $I_1 = \{human(marie\_curie)\}$ and $I_2 = \{thing(marie\_curie)\}$; letting instead $A = \{stands/1\}$, we get one interpretation $I_3 = \{stands(marie\_\ curie)\}$. For both choices of abducible predicates, this seems reasonable. However, including all predicates as abducibles, i.e., $A = \{human/1, thing/1, stands/1\}$, we obtain three minimal explanations, namely I_1, I_2 and I_3; the purely syntactic characterization of minimality (by a subset relationship) proves unsatisfactory, since I_3 subsumes both I_1 and I_2. Defining minimality by logical entailment is not satisfactory either, since for $A = \{human/1, thing/1, stands/1\}$ this would reduce to one minimal explanation, namely I_3, so that no explicit information about abducibles $thing/1$ and $human/1$ is provided.

The following definition seems to provide the intuitively right characterization independently of the sort of knowledge bases that are used.

Definition 6. *Let Γ be a constrained abductive grammar with constraint system $\langle A, K^{bg}\rangle$. An interpretation I of a string τ in Γ is called a* substantiated interpretation *whenever $I = M|_A$ where M is a model of $K^{bg} \cup I$ and $M|_A$ is the subset of literals in M whose predicate belongs to A. A substantiated interpretation is* minimal *whenever no proper subset of it is a substantiated interpretation.*

The following property indicates that substantiated interpretations characterize the full space of interpretations.

Proposition 1. *Let Γ be a constrained abductive grammar and I an interpretation of a string τ in Γ. Then there exists an substantiated interpretation E with $I \subseteq E$; when I is minimal, E can be chosen as a minimal substantiated interpretation.*

Example 6. Assume a constrained grammar similar to Γ_4 of ex. 5 above, but with the extended set of abducible predicates $\{human/1, thing/1, stands/1, thinks/1\}$. This provides two substantiated interpretations of *marie curie stands*:

$$\{stands(marie\_curie), thing(marie\_curie)\}$$
$$\{stands(marie\_curie), thinks(marie\_curie), human(marie\_curie)\}$$

5 Dynamic Abductive Grammars

Here we drop the restriction of a first-order semantics for the knowledge base and allow it to develop along a derivation (as opposed to the static abductive case). We allow now explicit "knowledge update terms" within grammar rules to modify the knowledge base whenever the rule is applied in a derivation.

We now let $\mathcal{K}$ be some set of possible formulas that can appear in grammar rules and refer to the knowledge base. Whenever $k \in \mathcal{K}$ and K is a knowledge base, we use the notation accommodate(k, K) for a new knowledge base updated with k. The actual definition of 'accommodate' depends on the sort of knowledge bases and updating operations $\mathcal{K}$ as well as the application. Intuitively, we may expect that accommodate(k, K) entails k and that K is repaired properly in order to restore consistency, if necessary. Notice that accommodate(k, K) may have many possible or no values at all; the latter corresponds to impossibility of accommodating k in any acceptable way. There is no need for an explicit constraint system as a similar effect is implied by accomodate.

Definition 7. *A* dynamic abuctive grammar *is a 5-tuple $\langle N, T, K^{init}, R, S\rangle$ where*

- *N are* nonterminal symbols*, each with a given arity, and N' as above,*
- *T a set of* terminal (symbol)s*,*
- *K^{init} is an initial* knowledge base*,*
- *R is a set of* rules*, each of the form*
  $N' \rightarrow (\{\mathcal{K}\} + N' + T)^*$.
- *S is a 0-ary nonterminal called the* start symbol.

Definition 8. *Given a dynamic abuctive grammar $\Gamma = \langle N, T, K^{init}, R, S\rangle$, the* derivation relation $\Rightarrow_\Gamma$ *is defined over pairs of sentential form and knowledge base by the following two cases.*

$$\langle \alpha n \beta, K\rangle \Rightarrow_\Gamma \langle \alpha\gamma\beta, K\rangle \quad \textit{whenever} \quad (n \rightarrow \gamma) \in \text{ground}(R)$$
$$\langle \alpha\{k\}\beta, K\rangle \Rightarrow_\Gamma \langle \alpha\beta, K'\rangle \quad \textit{whenever} \quad \text{accommodate}(k, K) = K'$$

Derivations and notation $\Rightarrow^*_\Gamma$ *are defined as usual. The* interpreted language *defined by* Γ, $L(\Gamma)$ *consists of all pairs of the form* $\langle \tau, K\rangle$ *for which* $\langle S, K_0\rangle \Rightarrow^*_\Gamma \langle \tau, K\rangle$ *and* $\tau \in T^*$.

When $\langle \tau, K\rangle \in L(\Gamma)$, *we say that* K *is an* interpretation *of* τ *by* Γ. *Minimal interpretations are defined as usual. The* LR (or left-to-right) *derivation relation, denoted* $\overset{LR}{\Rightarrow}_\Gamma$, *is defined in a similar way, except in the two cases requires* $\alpha \in T^*$. *PR interpretations and their minimality notions is defined analogously.*

Non-minimal interpretations of a given string may be created due to different choices of possible accommodations as well as alternative choice of rule instances. We have not defined the explicit notion of substantiated interpretation, but this will be relevant to study in detail for any specific choice of knowledge representation mechanism employed in a dynamic abductive grammar.

Constrained abductive interpretation problems were specified in an abstract way that does not anticipate any solution method. We can, as an important result, show a translation of a constrained abductive grammar into a version with a dynamic knowledge base, which will generate an interpretation as a side-effect of syntactic derivation.

Theorem 1. *Consider a constrained abductive grammar* $AG = \langle \Gamma, C\rangle$ *with* $\Gamma = \langle N, T, R, S\rangle$ *and* $C = \langle A, K^{bg}\rangle$. *Construct a dynamic abuctive grammar* $\Delta(AG) = \langle N, T, K^{bg}, R, S\rangle$ *by, for any* Σ *formula* k *and any knowledge base* K, *having the set of acceptable results for* accommodate(k, K) *being of the form* $K \cup K'$ *where* K' *is a smallest set of ground abducible literals such that* $K \cup K'$ *is consistent and* $K \cup K' \models k$; *if no such* K' *exists,* accommodate(k, K) *is not defined.*

Completeness: Then, for any minimal interpretation I *of a string* τ *under* AG, $K^{bg} \cup I$ *is an interpretation of* τ *under* $\Delta(AG)$.

Soundness: Any interpretation of a string τ *under* $\Delta(AG)$ *can be written as* $K^{bg} \cup I$ *where* I *is an interpretation of* τ *under* AG.

The proof is straightforward and omitted: a derivation in one grammar is easily mapped into a derivation of the other and vice versa. It is interesting to see that this theorem holds without any restrictions on the derivation order, which emphasizes why we called the traditional approach to abduction static.

The theorem showed how a static abductive analysis can be implemented using a suitable accommodation function with LR or unrestricted derivations. The more interesting applications are when the accomodation function is truly nonmonotonic[1] in which case it seems natural to confine derivations to LR: that relates the temporal information embedded in one sentence being uttered before others, and thus knowledge changes over time. The knowledge base at time t can thus be thought of as a current set of beliefs that apply to the state of affairs at time t and which may be inconsistent with the state of affairs at time $t + 1$.

[1] Static abduction is often mistaken as a kind of nonmonotonic reason since knowledge operationally is added piece by piece as, e.g., indicated by our theorem, but this does not change the fact the static abduction is searching for a single knowledge base that can explain the whole discourse.

Example 7. The framework of Global Abduction proposed by [12,13] uses a knowledge representation based on logic programs whose fact base may change dynamically; it has the distinguished feature that adding information contradicting previous knowledge does not lead to a failure, but the newer information replaces the old one. [14] has shown how this paradigm can be mapped into Constraint Handling Rules [8]. Global Abduction was intended for agent planning in a dynamic environment, but it will fit nicely as knowledge representation formalism with accommodation function within a dynamic abductive grammar.

The update mechanism used in Global Abduction provides a particularly straightforward way to handle variants of general belief revision [15]. It seems possible that any suggestion for a belief revision system can utilized in our framework.

6 Related Work and Concluding Remarks

Abductive Logic Grammars relate to *abductive logic programming* [16] by implementing knowledge base extraction through constrained-based abduction – a paradigm used, e.g., for abducing molecular acid strings from RNA secondary structure [17] – with the novel feature that they allow abducibles to appear in heads of clauses, cf. our notion of substantiated answers. They relate also to *knowledge extraction from texts*, with the novelty that they allow us to blend linguistic and meaning representation capabilities within the same process, so that their interaction can fine-tune the resulting knowledge bases in ways warranted by the linguistic information, thus going beyond the state of the art capabilities in text mining. And of course, they relate to *logic grammars*, with the novelty that meaning representations can be richer than ever, since they can interact during the process of their construction with all kinds of contextual information, gleaned this time through abduction, to guide that process.

Among specific related works, [18] is perhaps the closest to our own: it couples a syntactico-semantic grammar and a knowledge base implemented in Description Logics which is consulted by the grammar in order to ensure that only semantically acceptable parses are built. While the knowledge base can also learn new information from the parser's calls (i.e., from a new sentence being analysed), this information focusses mostly on lexical semantics, given that this approach's main aim is semantic correctness. In contrast, Abductive Logic Grammars can infer full knowledge bases from their description in human language, semantic correctness being only one of its possible applications. Future work will focus in concrete applications of this promising new framework, and on incorporating weighting schemes for prioritizing different interpretations, e.g., based on probabilities that have proved effective in other logic and constraint based settings for abduction [19,20,21].

References

1. Martin, P., Eklund, P.W.: Knowledge retrieval and the world wide web. IEEE Intelligent Systems 15(3), 18–25 (2000)
2. van Wijngaarden, A.: The generative power of two-level grammars. In: Loeckx, J. (ed.) ICALP 1974. LNCS, vol. 14, pp. 9–16. Springer, Heidelberg (1974)
3. Colmerauer, A.: Metamorphosis grammars. In: Bolc, L. (ed.) Natural Language Communication with Computers. LNCS, vol. 63, pp. 133–189. Springer, Heidelberg (1978)
4. Pereira, F.C.N., Warren, D.H.D.: Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. Artificial Intelligence 13(3), 231–278 (1980)
5. Hodas, J.S., Miller, D.: Logic programming in a fragment of intuitionistic linear logic. In: LICS, pp. 32–42. IEEE Computer Society Press, Los Alamitos (1991)
6. Dahl, V., Tarau, P., Li, R.: Assumption grammars for processing natural language. In: ICLP, pp. 256–270 (1997)
7. Christiansen, H.: CHR Grammars. Int'l Journal on Theory and Practice of Logic Programming 5(4-5), 467–501 (2005)
8. Frühwirth, T.: Theory and practice of constraint handling rules, special issue on constraint logic programming. Journal of Logic Programming 37(1–3), 95–138 (1998)
9. Christiansen, H., Dahl, V.: Meaning in Context. In: Dey, A.K., Kokinov, B., Leake, D.B., Turner, R. (eds.) CONTEXT 2005. LNCS, vol. 3554, pp. 97–111. Springer, Heidelberg (2005)
10. Christiansen, H., Dahl, V.: HYPROLOG: A new logic programming language with assumptions and abduction. In: Gabbrielli, M., Gupta, G. (eds.) ICLP 2005. LNCS, vol. 3668, pp. 159–173. Springer, Heidelberg (2005)
11. Lloyd, J.W.: Foundations of logic programming, 2nd extended edn. Springer, Heidelberg (1987)
12. Satoh, K.: "All's well that ends well" - a proposal of global abduction. In: Delgrande, J.P., Schaub, T. (eds.) NMR, pp. 360–367 (2004)
13. Satoh, K.: An application of global abduction to an information agent which modifies a plan upon failure - preliminary report. In: Leite, J., Torroni, P. (eds.) CLIMA 2004. LNCS, vol. 3487, pp. 213–229. Springer, Heidelberg (2005)
14. Christiansen, H.: On the implementation of global abduction. In: Inoue, K., Satoh, K., Toni, F. (eds.) CLIMA 2006. LNCS, vol. 4371, pp. 226–245. Springer, Heidelberg (2007)
15. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic 50(2), 510–530 (1985)
16. Kakas, A., Kowalski, R., Toni, F.: The role of abduction in logic programming. In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 5, pp. 235–324. Oxford University Press, Oxford (1998)
17. Bavarian, M., Dahl, V.: Constraint based methods for biological sequence analysis. Journal of Universal Computing Science 12(11), 1500–1520 (2006)

18. Sagot, B., Ghali, A.E.: Coupling grammar and knowledge base: Range concatenation grammars and description logics. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2004. LNCS, vol. 3206, pp. 195–202. Springer, Heidelberg (2004)
19. Poole, D.: Logic programming, abduction and probability - a top-down anytime algorithm for estimating prior and posterior probabilities. New Generation Computing 11(3), 377–400 (1993)
20. Sato, T., Kameya, Y.: Prism: A language for symbolic-statistical modeling. In: IJCAI, pp. 1330–1339 (1997)
21. Christiansen, H.: Implementing probabilistic abductive logic programming with constraint handling rules. In: Schrijvers, T., Frühwirth, T. (eds.) Constraint Handling Rules, Current Research Topics. LNCS (LNAI), vol. 5388, pp. 85–118. Springer, Heidelberg (2008)

On the Syntax-Semantics Interface: From Convergent Grammar to Abstract Categorial Grammar*

Philippe de Groote[1], Sylvain Pogodalla[2], and Carl Pollard[3]

[1] LORIA/INRIA Nancy – Grand Est
philippe.degroote@loria.fr
[2] LORIA/INRIA Nancy – Grand Est
sylvain.pogodalla@loria.fr
[3] The Ohio State University
pollard@ling.ohio-state.edu

Abstract. Cooper's storage technique for scoping *in situ* operators has been employed in theoretical and computational grammars of natural language (NL) for over thirty years, but has been widely viewed as ad hoc and unprincipled. Recent work by Pollard within the framework of convergent grammar (CVG) took a step in the direction of clarifying the logical status of Cooper storage by encoding its rules within an explicit but nonstandard natural deduction (ND) format. Here we provide further clarification by showing how to encode a CVG with storage within a logical grammar framework—abstract categorial grammar (ACG)—that utilizes no logical resources beyond those of standard linear deduction.

Introduction

A long-standing challenge for designers of NL grammar frameworks is posed by **in situ operators**, expressions such as quantified noun phrases (QNPs, e.g. *every linguist*), wh-expressions (e.g. *which linguist*), and comparative phrases (e.g. *more than five dollars*), whose semantic scope is underdetermined by their syntactic position. One family of approaches, employed by computational semanticists [1] and some versions of categorial grammar [2] and phrase structure grammar [3,4] employs the **storage** technique first proposed by Cooper [5]. In these approaches, syntactic and semantic derivations proceed in parallel, much as in classical Montague grammar (CMG [6]) except that sentences which differ only with respect to the scope of in-situ operators have identical syntactic derivations.[1] Where they differ is in the semantic derivations: the meaning of an in-situ operator is *stored* together with a copy of the variable that occupies the hole in a delimited semantic continuation over which the stored operator will

* The authors wish to acknowledge support from the Conseil Régional de Lorraine.

[1] In CMG, syntactic derivations for different scopings of a sentence differ with respect to the point from which a QNP is 'lowered' into the position of a syntactic variable.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 182–196, 2009.

scope when it is *retrieved*; ambiguity arises from nondeterminism with respect to the retrieval site.

Although storage is easily grasped on an intuitive level, it has resisted a clear and convincing logical characterization, and is routinely scorned by theoreticians as 'ad hoc', 'baroque', or 'unprincipled'. Recent work [7,8] within the CVG framework provided a partial clarification by encoding storage and retrieval rules within a somewhat nonstandard ND semantic calculus (Section 1). The aim of this paper is to provide a logical characterization of storage/retrieval free of nonstandard features. To that end, we provide an explicit transformation of CVG interface derivations (parallel syntax-semantic derivations) into a framework (ACG [9]) that employs no logical resources beyond those of standard (linear) natural deduction. Section 2 provides a preliminary conversion of CVG by showing how to re-express the storage and retrieval rules (respectively) by standard ND hypotheses and another rule already present in CVG (analogous to Gazdar's [10] rule for unbounded dependencies). Section 3 introduces the target framework ACG. And Sect. 4 describes the transformation of a (pre-converted) CVG into an ACG.

1 Convergent Grammar

A CVG for an NL consists of three term calculi for syntax, semantics, and the interface. The syntactic calculus is a kind of applicative multimodal categorial grammar, the semantic calculus is broadly similar to a standard typed lambda calculus, and the interface calculus recursively specifies which syntax-semantics term pairs belong to the NL.[2] Formal presentation of these calculi are given in Appendix A.

In the **syntactic calculus**, types are syntactic categories, constants (nonlogical axioms) are words (broadly construed to subsume phrasal affixes, including intonationally realized ones), and variables (assumptions) are traces (axiom schema T), corresponding to 'overt movement' in generative grammar. Terms are (candidate syntactic analyses of) words and phrases.

For simplicity, we take as our basic syntactic types *np* (noun phrase), *s* (nontopicalized sentence), and *t* (topicalized sentence). Flavors of implication correspond not to directionality (as in Lambek calculus) but to grammatical functions. Thus syntactic arguments are explicitly identitifed as subjects ($-\circ_{\mathrm{s}}$), complements ($-\circ_{\mathrm{c}}$), or hosts of phrasal affixes ($-\circ_{\mathrm{a}}$). Additionally, there is a ternary ('Gazdar') type constructor A^C_B for the category of 'overtly moved' phrases that bind an A-trace in a B, resulting in a C.

Contexts (left of the $\vdash$) in syntactic rules represent unbound traces. The elimination rules (flavors of modus ponens) for the implications, also called merges (M), combine 'heads' with their syntactic arguments. The elimination rule G for the Gazdar constructor implements Gazdar's ([10]) rule for discharging traces; thus G compiles in the effect of a hypothetical proof step (trace binding) immediately and obligatorily followed by the consumption of the resulting abstract

[2] To handle phonology, ignored here, a fourth calculus is needed; and then the interface specifies phonology/syntax/semantics triples.

by the 'overtly moved' phrase. G requires no introduction rule because it is only introduced by lexical items ('overt movement triggers' such as wh-expressions, or the prosodically realized topicalizer).

In the CVG **semantic calculus**, as in familiar semantic λ-calculi, terms correspond to meanings, constants to word meanings, and implication elimination to function application. But there is no λ-abstraction! Instead, binding of semantic variables is effected by either (1) a semantic 'twin' of the Gazdar rule, which binds the semantic variable corresponding to a trace by (the meaning of) the 'overtly moved' phrase; or (2) by the Responsibility (retrieval) rule (R), which binds the semantic variable that marks the argument position of a stored ('covertly moved') *in situ* operator. Correspondingly, there are two mechanisms for introducing semantic variables into derivations: (1) ordinary hypotheses, which are the semantic counterparts of ('overt movement') traces; and the Commitment (Cooper storage) rule (C), which replaces a semantic operator a of type A^C_B with a variable $x : A$ while placing a (subscripted by x) in the store (also called the *co-context*), written to the left of the $\dashv$ (called co-turnstile).

The CVG **interface calculus** recursively defines a relation between syntactic and semantic terms. Lexical items pair syntactic words with their meanings. Hypotheses pair a trace with a semantic variable and enter the pair into the context. The C rule leaves the syntax of an *in situ* operator unchanged while storing its meaning in the co-context. The implication elimination rules pair each (subject-, complement-, or affix-)flavored syntactic implication elimination rule with ordinary semantic implication elimination. The G rule simultaneously binds a trace by an 'overtly moved' syntactic operator and a semantic variable by the corresponding semantic operator. And the R rule leaves the syntax of the retrieval site unchanged while binding a 'committed' semantic variable by the retrieved semantic operator.

2 About the Commitment and Retrieve Rules

In the CVG semantic calculus, C and R are the only rules that make use of the store (co-context), and their logical status is not obvious. This section shows that they can actually be derived from the other rules, in particular from the G rule. Indeed, the derivation on the left can be replaced by the one on the right[3]:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\,\pi_1\\ \Gamma \vdash a : A^C_B \dashv \Delta\end{array}}{\Gamma \vdash x : A \dashv a_x : A^C_B, \Delta}\,\mathrm{C} \qquad \begin{array}{c}\vdots\,\pi_2\\ \Gamma, \Gamma' \vdash b : B \dashv a_x : A^C_B, \Delta', \Delta\end{array}}{\Gamma, \Gamma' \vdash a_x\, b : C \dashv \Delta', \Delta}\,\mathrm{R}$$

$$\rightsquigarrow \qquad \dfrac{\begin{array}{c}\vdots\,\pi_1\\ \Gamma \vdash a : A^C_B \dashv \Delta\end{array} \qquad \dfrac{}{x : A \vdash x : A \dashv}\ \begin{array}{c}\vdots\,\pi_2\\ x : A, \Gamma' \vdash b : B \dashv \Delta'\end{array}}{\Gamma, \Gamma' \vdash a_x\, b : C \dashv \Delta, \Delta'}\,\mathrm{G}$$

[3] The fact that we can divide the context into Γ and Γ' and the store into Δ and Δ', and that Γ and Δ are preserved, is shown in Proposition 1 of Appendix B.

This shows we can eliminate the store, resulting in a more traditional presentation of the underlying logical calculus. On the other hand, in the CVG interface calculus, this technique for elimiating C and R rules does not quite go through because the G rule requires both the syntactic type and the semantic type to be of the form α^{γ}_{β}. This difficulty is overcome by adding the following Shift rule to the interface calculus:

$$\frac{\Gamma \vdash a, b : A, B^D_C \dashv \Delta}{\Gamma \vdash \mathrm{S}_E\, a, b : A^E_E, B^D_C \dashv \Delta}\ \mathrm{Shift}_E$$

where S_E is a functional term whose application to an A produces a A^E_E. Then we can transform

$$\frac{\dfrac{\vdots \pi_1}{\Gamma \vdash a, b : A, B^D_C \dashv \Delta}\ \mathrm{C}}{\Gamma \vdash a, x : A, B \dashv b_x : B^D_C, \Delta}$$

$$\frac{\dfrac{\vdots \pi_2}{\Gamma, \Gamma' \vdash e, c : E, C \dashv b_x : B^D_C, \Delta, \Delta'}\ \mathrm{R}}{\Gamma, \Gamma' \vdash e, b_x\, c : E, D \dashv \Delta', \Delta}$$

to:

$$\frac{\dfrac{\dfrac{\vdots \pi_1}{\Gamma \vdash a, b : A, B^D_C \dashv \Delta}\ \mathrm{Shift}_E}{\Gamma \vdash \mathrm{S}_E\, a, b : A^E_E, B^D_C \dashv \Delta} \qquad \dfrac{\dfrac{}{t, x : A, B \vdash t, x : A, B \dashv}}{\vdots \pi_2}}{\Gamma, \Gamma' \vdash (\mathrm{S}_E a)_t\, e, b_x\, c : E, D \dashv \Delta, \Delta'}\ \mathrm{G}$$

provided $(\mathrm{S}_E a)_t\, e = (\mathrm{S}_E\, a)\, (\lambda t.e) = e[t := a]$. This follows from β-reduction as long as we take S_E to be $\lambda y\, P.P\, y$. Indeed:

$$(\mathrm{S}_E\, a)\, (\lambda t.e) = (\lambda y\, P.P\, y)\, a\, (\lambda t.e) =_\beta (\lambda P.P\, a)\, (\lambda t.e) =_\beta (\lambda t.e)\, a =_\beta e[t := a]$$

With this additional construct, we can get rid of the C and R rules in the CVG interface calculus. This construct is used in Section 4 to encode CVG into ACG. It can be seen as a rational reconstruction of Montague's quantifier lowering technique as nothing more than β-reduction in the syntax (unavailable to Montague since his syntactic calculus was purely applicative).

3 Abstract Categorial Grammar

Motivations. Abstract Categorial Grammars (ACGs) [9], which derive from type-theoretic grammars in the tradition of Lambek [11], Curry [12], and Montague [6], provide a framework in which several grammatical formalisms may be encoded [13]. The definition of an ACG is based on a small set of mathematical primitives from type-theory, λ-calculus, and linear logic. These primitives combine via simple composition rules, which offers ACGs a good flexibility. In

particular, ACGs generate languages of linear λ-terms, which generalizes both string and tree languages. They also provide the user direct control over the parse structures of the grammar, which allows several grammatical architectures to be defined in terms of ACG.

Mathematical preliminaries. Let A be a finite set of atomic types, and let $\mathscr{T}_A$ be the set of linear functional types types (in notation, $\alpha \multimap \beta$) built upon A. A *higher-order linear signature* is then defined to be a triple $\Sigma = \langle A, C, \tau \rangle$, where: A is a finite set of atomic types; C is a finite set of constants; and τ is a mapping from C to $\mathscr{T}_A$. A higher-order linear signature will also be called a *vocabulary*. In the sequel, we will write A_Σ, C_Σ, and τ_Σ to designate the three components of a signature Σ, and we will write $\mathscr{T}_\Sigma$ for $\mathscr{T}_{A_\Sigma}$.

We take for granted the definition of a λ-term, and we let the relation of $\beta\eta$-conversion to be the notion of equality between λ-terms. Given a higher-order signature Σ, we write Λ_Σ for the set of *linear simply-typed λ-terms*.

Let Σ and Ξ be two higher-order linear signatures. A *lexicon* $\mathscr{L}$ from Σ to Ξ (in notation, $\mathscr{L} : \Sigma \longrightarrow \Xi$) is defined to be a pair $\mathscr{L} = \langle \eta, \theta \rangle$ such that: η is a mapping from A_Σ into $\mathscr{T}_\Xi$; θ is a mapping from C_Σ into Λ_Ξ; and for every $c \in C_\Sigma$, the following typing judgement is derivable: $\vdash_\Xi \theta(c) : \hat{\eta}(\tau_\Sigma(c))$, where $\hat{\eta} : \mathscr{T}_\Sigma \to \mathscr{T}_\Xi$ is the unique homomorphic extension of η.[4]

Let $\hat{\theta} : \Lambda_\Sigma \to \Lambda_\Xi$ be the unique λ-term homomorphism that extends θ.[5] We will use $\mathscr{L}$ to denote both $\hat{\eta}$ and $\hat{\theta}$, the intended meaning being clear from the context. When Γ denotes a typing environment '$x_1 : \alpha_1, \ldots, x_n : \alpha_n$', we will write $\mathscr{L}(\Gamma)$ for '$x_1 : \mathscr{L}(\alpha_1), \ldots, x_n : \mathscr{L}(\alpha_n)$'. Using these notations, we have that the last condition for $\mathscr{L}$ induces the following property: if $\Gamma \vdash_\Sigma t : \alpha$ then $\mathscr{L}(\Gamma) \vdash_\Xi \mathscr{L}(t) : \mathscr{L}(\alpha)$.

Definition 1. *An* abstract categorial grammar *is a quadruple $\mathscr{G} = \langle \Sigma, \Xi, \mathscr{L}, s \rangle$ where:*

1. *Σ and Ξ are two higher-order linear signatures, which are called the* abstract vocabulary *and the* object vocabulary*, respectively;*
2. *$\mathscr{L} : \Sigma \longrightarrow \Xi$ is a lexicon from the abstract vocabulary to the object vocabulary;*
3. *$s \in \mathscr{T}_\Sigma$ is a type of the abstract vocabulary, which is called the* distinguished type *of the grammar.*

A possible intuition behind this definition is that the object vocabulary specifies the surface structures of the grammars, the abstract vocabulary specifies its abstract parse structures, and the lexicon specifies how to map abstract parse structures to surface structures. As for the distinguished type, it plays the same part as the start symbol of the phrase structures grammars. This motivates the following definitions.

The *abstract language* of an ACG is the set of closed linear λ-terms that are built on the abstract vocabulary, and whose type is the distinguished type:

[4] That is $\hat{\eta}(a) = \eta(a)$ and $\hat{\eta}(\alpha \multimap \beta) = \hat{\eta}(\alpha) \multimap \hat{\eta}(\beta)$.

[5] That is $\hat{\theta}(c) = \theta(c)$, $\hat{\theta}(x) = x$, $\hat{\theta}(\lambda x.\, t) = \lambda x.\, \hat{\theta}(t)$, and $\hat{\theta}(t\, u) = \hat{\theta}(t)\, \hat{\theta}(u)$.

$$\mathcal{A}(\mathscr{G}) = \{t \in \Lambda_\Sigma \mid \vdash_\Sigma t : s \text{ is derivable}\}$$

On the other hand, the object language of the grammar is defined to be the image of its abstract language by the lexicon:

$$\mathcal{O}(\mathscr{G}) = \{t \in \Lambda_\Xi \mid \exists u \in \mathcal{A}(\mathscr{G}).\ t = \mathscr{L}(u)\}$$

It is important to note that, from a purely mathematical point of view, there is no structural difference between the abstract and the object vocabulary: both are higher-order signatures. Consequently, the intuition we have given above is only a possible interpretation of the definition, and one may conceive other possible grammatical architectures. Such an architecture consists of two ACGs sharing the same abstract vocabulary, the object vocabulary of the first ACG corresponding to the syntactic structures of the grammar, and the one of the second ACG corresponding to the semantic structures of the grammar. Then, the common abstract vocabulary corresponds to the transfer structures of the syntax/semantics interface. This is precisely the architecture that the next section will exemplify.

4 ACG Encoding of CVG

The Overall Architecture. As Section 1 shows, whether a pair of a syntactic term and a semantic term belongs to the language depends on whether it is derivable from the lexicon in the CVG interface calculus. Such a pair is indeed an *(interface) proof term* corresponding to the derivation. So the first step towards the encoding of CVG into ACG is to provide an abstract language that generates the same proof terms as those of the CVG interface. For a given CVG G, we shall call $\Sigma_{I(G)}$ the higher-order signature that will generate the same proof terms as G. Then, any ACG whose abstract vocabulary is $\Sigma_{I(G)}$ will generate these proof terms. And indeed we will use two ACG sharing this abstract vocabulary to map the (interface) proof terms into syntactic terms and into semantic terms respectively. So we need two other signatures: one allowing us to express the syntactic terms, which we call $\Sigma_{\text{SimpleSyn}(G)}$, and another allowing us to express the semantic terms, which we call $\Sigma_{\text{Log}(G)}$.

Finally, we need to be able to recover the two components of the pair out of the proof term of the interface calculus. This means having two ACG sharing the same abstract language (the closed terms of $\Lambda(\Sigma_{I(G)})$ of some distinguished type) and whose object vocabularies are respectively $\Sigma_{\text{SimpleSyn}(G)}$ and $\Sigma_{\text{Log}(G)}$. Fig. 1 illustrates the architecture with $\mathscr{G}_{\text{Syn}} = \langle \Sigma_{I(G)}, \Sigma_{\text{SimpleSyn}(G)}, \mathscr{L}_{\text{Syn}}, s\rangle$ the first ACG that encodes the mapping from interface proof terms to syntactic terms, and $\mathscr{G}_{\text{Sem}} = \langle \Sigma_{I(G)}, \Sigma_{\text{Log}(G)}, \mathscr{L}_{\text{Log}}, s\rangle$ the second ACG that encodes the mapping from interface proof terms to semantic formulas. It should be clear that this architecture can be extended so as to get phonological forms and conventional logical forms (say, in TY_2) using similar techniques. The latter requires non-linear λ-terms, an extension already available to ACG [14] . So we focus here on the (simple) syntax-semantics interface only, which requires only linear terms.

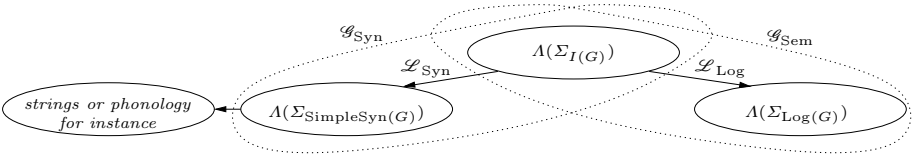


Fig. 1. Overall architecture of the ACG encoding of a CVG

Table 1. CVG lexicon for topicalization

Chris, Chris' :	np, ι	top, top' :	$np \multimap_{\text{a}} np^t_s, \iota \multimap \iota^\pi_\pi$
liked, like' :	$np \multimap_{\text{c}} np \multimap_{\text{s}} s, \iota \multimap \iota \multimap \pi$	$\text{top}_{\text{in-situ}}$, top' :	$np \multimap_{\text{a}} np, \iota \multimap \iota^\pi_\pi$

Table 2. ACG translation of the CVG lexicon for topicalization

CHRIS :	$\langle np, \iota\rangle$	TOP :	$\langle np, \iota\rangle \multimap \langle np^t_s, \iota^\pi_\pi\rangle$
LIKED :	$\langle np, \iota\rangle \multimap \langle np, \iota\rangle \multimap \langle s, \pi\rangle$	$\text{TOP}_{\text{IN-SITU}}$:	$\langle np, \iota\rangle \multimap \langle np, \iota^\pi_\pi\rangle$

We begin by providing an example of a CVG lexicon (Table 1). Recall that the syntactic type t is for overtly topicalized sentences, and $\multimap a$ is the flavor of implication for affixation. We recursively define the translation $\overline{\cdot}^\tau$ of CVG pairs of syntactic and semantics types to $\Sigma_{I(G)}$ as:

- $\overline{\alpha, \beta}^\tau = \langle \alpha, \beta\rangle$ if either α or β is atomic or of the form γ^ϵ_δ. Note that this new type $\langle a, \beta\rangle$ is an *atomic* type of $\Sigma_{I(G)}$;
- $\overline{\alpha \multimap \beta, \alpha' \multimap \beta'}^\tau = \overline{\alpha, \alpha'}^\tau \multimap \overline{\beta, \beta'}^\tau$[6].

When ranging over the set of types provided by the CVG lexicon[7], we get all the atomic types of $\Sigma_{I(G)}$. Then, for any $w, f : \alpha, \beta$ of the CVG lexicon of G, we add the constant $\overline{w, f}^{\text{c}} = \text{w}$ of type $\overline{\alpha, \beta}^\tau$ to the signature $\Sigma_{I(G)}$.

The application of $\overline{\cdot}^{\text{c}}$ and $\overline{\cdot}^\tau$ to the lexicon of Table 1 yields the signature $\Sigma_{I(G)}$ of Table 2. Being able to use the constants associated to the topicalization operators in building new terms requires additional constants having e.g. $\langle np, \iota^\pi_\pi\rangle$ as parameters. We delay this construct to Sect. 4.

Constants and types in $\Sigma_{\text{SimpleSyn}(G)}$ and $\Sigma_{\text{Log}(G)}$ simply reflect that we want them to build terms in the syntax and in the semantics respectively. First, note that a term of type α^γ_β, according to the CVG rules, can be applied to a term of type $\alpha \multimap \beta$ to return a term of type γ. Moreover, the type α^γ_β does not exist in any of the ACG object vocabularies. Hence we recursively define the $[\![\cdot]\!]$

[6] This translation preserves the order of the types. Hence, in the ACG settings, it allows abstraction everywhere. This does not fulfill one of the CVG requirements. However, since it is always possible from an ACG $\mathscr{G}$ to build a new ACG $\mathscr{G}'$ such that $\mathcal{O}(\mathscr{G}') = \{t \in \mathcal{A}(\mathscr{G}) | t \text{ consists only in applications}\}$ (see the construct in Appendix C), we can assume without loss of generality that we here deal only with second order terms.

[7] Actually, we should also consider additional types issuing from types of the form α^γ_β when one of the α, β or γ is itself a type of this form.

function that turns CVG syntactic and semantic types into linear types (as used in higher-order signatures) as:

- $[\![a]\!] = a$ if a is atomic
- $[\![\alpha^{\gamma}_{\beta}]\!] = ([\![\alpha]\!] \multimap [\![\beta]\!]) \multimap [\![\gamma]\!]$
- $[\![\alpha \multimap_{\mathrm{x}} \beta]\!] = [\![\alpha]\!] \multimap [\![\beta]\!]$

Then, for any CVG constant $w, f : \alpha, \beta$ we have $\overline{w, f}^{\mathrm{c}} = \mathrm{w} : \overline{\alpha, \beta}^{\tau}$ in $\Sigma_{I(G)}$:

$$\begin{array}{llll} \mathscr{L}_{\mathrm{Syn}}(\mathrm{w}) & = & w & \mathscr{L}_{\mathrm{Log}}(\mathrm{w}) & = & f \\ \mathscr{L}_{\mathrm{Syn}}(\overline{\alpha, \beta}^{\tau}) & = & [\![\alpha]\!] & \mathscr{L}_{\mathrm{Log}}(\overline{\alpha, \beta}^{\tau}) & = & [\![\beta]\!] \end{array}$$

So the lexicon of Table 1 gives[8]:

$$\begin{array}{ll} \mathscr{L}_{\mathrm{Syn}}(\textsc{Chris}) = \mathsf{Chris} & \mathscr{L}_{\mathrm{Syn}}(\textsc{liked}) = \lambda xy.\left[^{\mathrm{s}} y \ \left[\mathsf{liked}\ x^{\,\mathrm{c}}\right]\right] \\ \mathscr{L}_{\mathrm{Log}}(\textsc{Chris}) = \mathsf{Chris'} & \mathscr{L}_{\mathrm{Log}}(\textsc{liked}) = \lambda xy.\mathsf{like'}\, y\, x \end{array}$$

And we get the trivial translations:

$$\begin{array}{l} \mathscr{L}_{\mathrm{Syn}}(\textsc{liked Sandy Chris}) = \left[^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked\ Sandy}^{\,\mathrm{c}}\right]\right] : s \\ \mathscr{L}_{\mathrm{Log}}(\textsc{liked Sandy Chris}) = \mathsf{like'\ Chris'\ Sandy'} : \pi \end{array}$$

On the Encoding of CVG Rules. There is a trivial one-to-one mapping between the CVG rules Lexicon, Trace, and Subject and Complement Modus Ponens, and the standard typing rules of linear λ-calculus of ACG: constant typing rule (non logical axiom), identity rule and application. So the ACG derivation that proves $\vdash_{\Sigma_{I(G)}} \textsc{liked Sandy Chris} : \langle s, \pi \rangle$ in $\Lambda(\Sigma_{I(G)})$ is isomorphic to $\vdash \left[^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked\ Sandy}^{\,\mathrm{c}}\right]\right], \mathsf{like'\ Sandy'\ Chris'} : s, \pi \dashv$ as a CVG interface derivation. But the CVG G rule has no counterpart in the ACG type system. So it needs to be introduced using constants in $\Sigma_{I(G)}$.

Let's assume a CVG derivation using the following rule:

$$\frac{\begin{array}{cc} \vdots \pi_1 & \vdots \pi_2 \\ \Gamma \vdash a, d : A^C_B, D^F_E \dashv \Delta & t, x : A, D; \Gamma' \vdash b, e : B, E \dashv \Delta' \end{array}}{\Gamma; \Gamma' \vdash a_t\, b, d_x\, e : C, F \dashv \Delta; \Delta'}\ \mathrm{G}$$

and that we are able to build two terms (or two ACG derivations) $\mathrm{T}_1 : \langle A^C_B, D^F_E \rangle$ and $\mathrm{T}_2 : \overline{B, E}^{\tau}$ of $\Lambda(\Sigma_{I(G)})$ corresponding to the two CVG derivations π_1 and π_2. Then, adding a constant $G_{\langle A^C_B, D^F_E \rangle}$ of type $\langle A^C_B, D^F_E \rangle \multimap (\overline{A, D}^{\tau} \multimap \overline{B, E}^{\tau}) \multimap \overline{C, F}^{\tau}$ in $\Sigma_{I(G)}$, we can build a new term $G_{\langle A^C_B, D^F_E \rangle}\, \mathrm{T}_1\, (\lambda y.\mathrm{T}_2) : \overline{C, F}^{\tau} \in \Lambda(\Sigma_{I(G)})$. It is then up to the lexicons to provide the good realizations of

[8] In order to help recognizing the CVG syntactic forms, we use additional operators of arity 2 in $\Sigma_{\mathrm{SimpleSyn}(G)}$: $\left[^{\mathrm{s}} s\ p\right]$ instead of writing $(p\, s)$ when p is of type $\alpha \multimap_{\mathrm{s}} \beta$ and $\left[p\ c^{\mathrm{x}}\right]$ instead of just $(p\, c)$ when p is of type $\alpha \multimap_{\mathrm{x}} \beta$ with $x \neq s$. This syntactic sugar is not sufficient to model the different flavors of the implication in CVG, the latter topic being beyond the scope of this paper.

$G_{\langle A^C_B, D^F_E\rangle}$ so that if $\mathscr{L}_{\text{Syn}}(\text{T}_1) = a$, $\mathscr{L}_{\text{Log}}(\text{T}_1) = d$, $\mathscr{L}_{\text{Syn}}(\text{T}_2) = b$ and $\mathscr{L}_{\text{Log}}(\text{T}_2) = e$ then $\mathscr{L}_{\text{Syn}}(G_{\langle A^C_B, D^F_E\rangle}\ \text{T}_1\ (\lambda y.\text{T}_2)) = a\,(\lambda y.b)$ and $\mathscr{L}_{\text{Log}}(G_{\langle A^C_B, D^F_E\rangle}\ \text{T}_1\ (\lambda y.\text{T}_2)) = d\,(\lambda y.e)$. This is realized when $\mathscr{L}_{\text{Syn}}(G_{\langle A^C_B, D^F_E\rangle}) = \mathscr{L}_{\text{Log}}(G_{\langle A^C_B, D^F_E\rangle}) = \lambda Q\,R.Q\,R$. A CVG derivation using the (not in-situ) topicalization lexical item and the G rule from $\vdash \left[\mathsf{Sandy\ top}^{\,\mathrm{a}}\right], \mathsf{top'\,Sandy'} : np^t_s, \iota^\pi_\pi \dashv$ and from $t, x : np, \iota \vdash \left[^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right], \mathsf{like'}\ x\ \mathsf{Chris'} : s, \pi \dashv$ would result (conclusion of a G rule) in a proof of $\vdash \left[\mathsf{Sandy\ top}^{\,\mathrm{a}}\right]_t \left[^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right], (\mathsf{top'\,Sandy'})_x\,(\mathsf{like'}\ x\ \mathsf{Chris'}) : t, \pi \dashv$, the latter being isomorphic to the derivation in $\Lambda(\Sigma_{I(G)})$ proving: $\vdash_{\Sigma_{I(G)}} G_{\langle np^t_s, \iota^\pi_\pi\rangle}(\text{TOP SANDY})(\lambda x.\text{LIKED}\ x\ \text{CHRIS}) : \langle t, \pi\rangle$. Let's call this term T. Then with $\mathscr{L}_{\text{Syn}}(\text{TOP}) = \lambda x.\left[\mathsf{top}\ x^{\,\mathrm{a}}\right] : [\![np \multimap_{\mathrm{a}} np^t_s]\!] = np \multimap (np \multimap s) \multimap t$, $\mathscr{L}_{\text{Log}}(\text{TOP}) = \mathsf{top'} : [\![\iota \multimap \iota^\pi_\pi]\!] = \iota \multimap (\iota \multimap \pi) \multimap \pi$, and $\mathscr{L}_{\text{Syn}}(G_{\langle np^t_s, \iota^\pi_\pi\rangle}) = \mathscr{L}_{\text{Log}}(G_{\langle np^t_s, \iota^\pi_\pi\rangle}) = \lambda P\,Q.P\,Q$, we have the expected result:

$$\mathscr{L}_{\text{Syn}}(t) = \left[\mathsf{Sandy\ top}^{\,\mathrm{a}}\right](\lambda x.\left[^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ x^{\,\mathrm{c}}\right]\right])$$
$$\mathscr{L}_{\text{Log}}(t) = (\mathsf{top'\,Sandy'})(\lambda x.\mathsf{like'}\ x\ \mathsf{Chris'})$$

The C and R Rules. Section 2 shows how we can get rid of the C and R rules in CVG derivations. It brings into play an additional Shift rule and an additional operator S. It should be clear from the previous section that we could add an abstract constant corresponding to this Shift rule. The main point is that its realization in the syntactic calculus by $\mathscr{L}_{\text{Syn}}$ should be $\text{S} = \lambda e\,P.P\,e$ and its realization in the semantics by $\mathscr{L}_{\text{Log}}$ should be the identity.

Technically, it would amount to have a new constant $\text{S}_{\langle A, B^D_C\rangle} : \langle a, B^D_C\rangle \multimap \langle A^E_E, B^D_C\rangle$ such that $\mathscr{L}_{\text{Log}}(\text{S}_{\langle A, B^D_C\rangle}) = \lambda x.x : [\![B^D_C]\!] \multimap [\![B^D_C]\!]$ (this rule does not change the semantics) and $\mathscr{L}_{\text{Syn}}(\text{S}_{\langle A, B^D_C\rangle}) = \lambda x\,P.P\,x : [\![A]\!] \multimap ([\![A]\!] \multimap [\![E]\!]) \multimap [\![E]\!]$ (this rule shift the syntactic type). But since this Shift rule is meant to occur together with a G rule to model C and R, the kind of term we will actually consider is: $t = G_{\langle A^E_E, B^D_C\rangle}(\text{S}_{\langle A, B^D_C\rangle}\ x)\,Q$ for some $x : \langle A, B^D_C\rangle$ and $Q : \langle A^E_E E, B^D_C\rangle$. And the interpretations of t in the syntactic and in the semantic calculus are:

$$\begin{aligned}\mathscr{L}_{\text{Log}}(t) &= (\lambda P\,Q.P\,Q)\\ &\quad((\lambda y.y)\mathscr{L}_{\text{Log}}(x))\mathscr{L}_{\text{Log}}(Q)\\ &= \mathscr{L}_{\text{Log}}(x)\,\mathscr{L}_{\text{Log}}(Q)\end{aligned} \qquad \begin{aligned}\mathscr{L}_{\text{Syn}}(t) &= (\lambda P\,Q.P\,Q)\\ &\quad((\lambda e P.P\,e)\mathscr{L}_{\text{Syn}}(x))\mathscr{L}_{\text{Syn}}(Q)\\ &= \mathscr{L}_{\text{Syn}}(Q)\,\mathscr{L}_{\text{Syn}}(x)\end{aligned}$$

So basically, $\mathscr{L}_{\text{Log}}(\lambda x\,Q.t) = \mathscr{L}_{\text{Log}}(G_{\langle A^E_E E, B^D_C\rangle})$, and this expresses that nothing new happens on the semantic side, while $\mathscr{L}_{\text{Syn}}(\lambda x\,Q.t) = \lambda x\,Q.Q\,x$ expresses that, somehow, the application is reversed on the syntactic side.

Rather than adding these new constants S (for each type), we integrate their interpretation into the associated G constant[9]. This amounts to compiling the composition of the two terms. So if we have a pair of type A, B^D_C occurring in a CVG G, we add to $\Sigma_{I(G)}$ a new constant $G^{\text{S}}_{\langle A, B^D_C\rangle} : \langle A, B^D_C\rangle \multimap (\overline{\langle A, B\rangle}^{\,\tau} \multimap$

[9] It correspond to the requirement that the Shift rule occurs just before the G rule in the modeling the interface C and R rule with the the G rule.

$\overline{\langle E, C\rangle}^{\tau}) \multimap \overline{\langle E, D\rangle}^{\tau}$ (basically the above term t) whose interpretations are: $\mathscr{L}_{\mathrm{Syn}}(G^{\mathrm{S}}_{\langle A, B^D_C\rangle}) = \lambda P\, Q.Q\, P$ and $\mathscr{L}_{\mathrm{Syn}}(G^{\mathrm{S}}_{\langle A, B^D_C\rangle}) = \lambda P\, Q.P\, Q$.

For instance, if we now use the in-situ topicalizer of Table 1 (triggered by stress for instance), from $\vdash \mathrm{S}_S\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right], \mathsf{top'}\ \mathsf{Sandy'} : np^S_S, \iota^{\pi}_{\pi} \dashv$ and $t, x : np, \iota \vdash \left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right], \mathsf{like'}\ x\ \mathsf{Chris'} : s, \pi \dashv$ we can derive, using the G rule, $\vdash (\mathrm{S}_S\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right])_t \left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right], (\mathsf{top'}\ \mathsf{Sandy'})_x(\mathsf{like'}\ x\ \mathsf{Chris'}) : s, \pi \dashv$ Note that:

$$\begin{aligned}(\mathrm{S}_S\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right])_t(\left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right]) &= ((\lambda e\, P.P\, e)\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right])\\ &\quad (\lambda t.\left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,\mathrm{c}}\right]\right])\\ &=_{\beta} \left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right]^{\mathrm{c}}\right]\right]\end{aligned}$$

In order to map this derivation to an ACG term, we use the constant $\textsc{top}_{\text{IN-SITU}} : \langle np, \iota\rangle \multimap \langle np, \iota^{\pi}_{\pi}\rangle$ and the constant that will simulate the G rule and the Shift rule together $G^{\mathrm{S}}_{\langle np, \iota^{\pi}_{\pi}\rangle} : \langle np, \iota^{\pi}_{\pi}\rangle \multimap (\langle np, \iota\rangle \multimap \langle s, \pi\rangle) \multimap \langle s, \pi\rangle$ such that, according to what precedes: $\mathscr{L}_{\mathrm{Syn}}(G^{\mathrm{S}}_{\langle np, \iota^{\pi}_{\pi}\rangle}) = \lambda P\, Q.Q\, P$ and $\mathscr{L}_{\mathrm{Log}}(G^{\mathrm{S}}_{\langle np, \iota^{\pi}_{\pi}\rangle}) = \lambda P\, Q.P\, Q$. Then the previous CVG derivation corresponds to the following term of $\Lambda(\Sigma_{I(G)})$: $t = G^{\mathrm{S}}_{\langle np, \iota^{\pi}_{\pi}\rangle}(\textsc{top}_{\text{IN-SITU}}\ \textsc{Sandy})(\lambda x.\textsc{liked}\ x\ \textsc{Chris})$ and its expected realizations as syntactic and semantic terms are:

$$\begin{aligned}\mathscr{L}_{\mathrm{Syn}}(t) &= (\lambda P\, Q.Q\, P)(\left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right])\ (\lambda x.\left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ x^{\,\mathrm{c}}\right]\right])\\ &= \left[{}^{\mathrm{s}}\mathsf{Chris}\ \left[\mathsf{liked}\ \left[\mathsf{Sandy}\ \mathsf{top}_{\text{in-situ}}{}^{\mathrm{a}}\right]^{\mathrm{c}}\right]\right]\end{aligned} \qquad \begin{aligned}\mathscr{L}_{\mathrm{Log}}(t) &= (\lambda P\, Q.P\, Q)(\mathsf{top'}\ \mathsf{Sandy'})\\ &\quad (\lambda x, \mathsf{like'}\ x\ \mathsf{Chris'})\\ &= (\mathsf{top'}\ \mathsf{Sandy'})(\lambda x.\mathsf{like'}\ x\ \mathsf{Chris'})\end{aligned}$$

Finally the $G_{\langle\alpha,\beta\rangle}$ and $G^{\mathrm{S}}_{\langle\alpha,\beta\rangle}$ are the only constants of the abstract signature having higher-order types. Hence, they are the only ones that will possibly trigger abstractions, fulfilling the CVG requirement.

When used in quantifier modeling, ambiguities are dealt with in CVG by the non determinism of the order in which semantic operators are retrieved from the store. It corresponds to the (reverse) order in which their ACG encoding are applied in the final term. However, by themselves, both accounts don't provide control on this order. Hence, when several quantifiers occur in the same sentence, all the relative orders of the quantifiers are possible.

Conclusion

We have shown how to encode a linguistically motivated *parallel* formalism, CVG, into a framework, ACG, that has mainly been used to encode syntactocentric formalisms until now. In addition to providing a logical basis for the CVG store mechanism, this encoding also sheds light on the various components (such as higher-order signatures) that are used in the interface calculus. It is noteworthy that the signature used to generate the interface proof terms relate to what is usually called *syntax* in mainstream categorial grammar, whereas the CVG *simple syntax* calculus is not expressed in such frameworks (while it can be using ACG, see [15]).

References

1. Blackburn, P., Bos, J.: Representation and Inference for Natural Language. A First Course in Computational Semantics. CSLI (2005)
2. Bach, E., Partee, B.H.: Anaphora and semantic structure (1980); Reprinted in Partee, B.H., Compositionality in Formal Semantics, pp. 122–152. Blackwell
3. Cooper, R.: Quantification and Syntactic Theory. Reidel, Dordrecht (1983)
4. Pollard, C., Sag, I.A.: Head-Driven Phrase Structure Grammar. CSLI Publications, Stanford (1994); Distributed by University of Chicago Press
5. Cooper, R.: Montague's Semantic Theory and Transformational Syntax. PhD thesis, University of Massachusetts at Amherst (1975)
6. Montague, R.: The proper treatment of quantification in ordinary english. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to natural language: proceedings of the 1970 Stanford workshop on Grammar and Semantics. Reidel, Dordrecht (1973)
7. Pollard, C.: Covert movement in logical grammar (submitted)
8. Pollard, C.: The calculus of responsibility and commitment (submitted)
9. de Groote, P.: Towards abstract categorial grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference, pp. 148–155 (2001)
10. Gazdar, G.: Unbounded dependencies and coordinate structure. Linguistic Inquiry 12, 155–184 (1981)
11. Lambek, J.: The mathematics of sentence structure. Amer. Math. Monthly 65, 154–170 (1958)
12. Curry, H.: Some logical aspects of grammatical structure. In: Jakobson, R. (ed.) Studies of Language and its Mathematical Aspects, Providence, Proc. of the 12th Symp. Appl. Math., pp. 56–68 (1961)
13. de Groote, P., Pogodalla, S.: On the expressive power of abstract categorial grammars: Representing context-free formalisms. Journal of Logic, Language and Information 13(4), 421–438 (2004), http://hal.inria.fr/inria-00112956/fr/
14. de Groote, P., Maarek, S.: Type-theoretic extensions of abstract categorial grammars. In: New Directions in Type-Theoretic Grammars, proceedings of the workshop, pp. 18–30 (2007), http://let.uvt.nl/general/people/rmuskens/ndttg/ndttg2007.pdf
15. Pogodalla, S.: Generalizing a proof-theoretic account of scope ambiguity. In: Geertzen, J., Thijsse, E., Bunt, H., Schiffrin, A. (eds.) Proceedings of the 7th International Workshop on Computational Semantics - IWCS-7, Tilburg University, Deparment of Communication and Information Sciences, pp. 154–165 (2007), http://hal.inria.fr/inria-00112898
16. Hinderer, S.: Automatisation de la construction sémantique dans TYn. PhD thesis, Université Henri Poincaré – Nancy 1 (2008)

A The CVG Calculi

A.1 The CVG Syntactic Calculus

$$\frac{}{\vdash a : A}\ \text{Lex} \qquad \frac{}{t : A \vdash t : A}\ \text{T } (t \text{ fresh})$$

$$\frac{\Gamma \vdash b : A \multimap_{\mathrm{s}} B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash \left[^{\mathrm{s}}\, a\ b\right] : B}\ \mathrm{M}_s \qquad \frac{\Gamma \vdash b : A \multimap_{\mathrm{c}} B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash \left[b\ a\,^{\mathrm{c}}\right] : B}\ \mathrm{M}_c$$

$$\frac{\Gamma \vdash b : A \multimap_{\mathrm{a}} B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash \left[b\ a\,^{\mathrm{a}}\right] : B}\ \mathrm{M}_a$$

$$\frac{\Gamma \vdash a : A^C_B \quad t : A; \Gamma' \vdash b : B}{\Gamma; \Gamma' \vdash a_t\, b : C}\ \mathrm{G}$$

A.2 The CVG Semantic Calculus

$$\frac{}{\vdash a : A \dashv}\ \text{Lex} \qquad \frac{}{x : B \vdash x : B \dashv}\ \text{T } (x \text{ fresh})$$

$$\frac{\vdash f : A \multimap B \dashv \Delta \quad \vdash a : A \dashv \Delta'}{\vdash (f\, a) : B \dashv \Delta; \Delta'}\ \mathrm{M}$$

$$\frac{\Gamma \vdash a : A^C_B \dashv \Delta \quad x : A; \Gamma' \vdash b : B \dashv \Delta'}{\Gamma; \Gamma' \vdash a_x\, b : C \dashv \Delta; \Delta'}\ \mathrm{G}$$

$$\frac{\vdash a : A^C_B \dashv \Delta}{\vdash x : A \dashv a_x : A^C_B; \Delta}\ \mathrm{C}\ (x \text{ fresh}) \qquad \frac{\vdash b : B \dashv a_x : A^C_B; \Delta}{\Gamma \vdash (a_x\, b) : C \dashv \Delta}\ \mathrm{R}$$

A.3 The CVG Interface Calculus

$$\frac{}{\vdash w, c : A, B \dashv}\ \text{Lex} \qquad \frac{}{x, t : A, B \vdash x, t : A, B \dashv}\ \text{T}$$

$$\frac{\Gamma \vdash f, v : A \multimap_{\mathrm{s}} B, C \multimap D \dashv \Delta \quad \Gamma' \vdash a, c : A, C \dashv \Delta'}{\Gamma; \Gamma' \vdash \left[^{\mathrm{s}}\, a\ f\right], (v\, c) : B, D \dashv \Delta; \Delta'}\ \mathrm{M}_s$$

$$\frac{\Gamma \vdash f, v : A \multimap_{\mathrm{c}} B, C \multimap D \dashv \Delta \quad \Gamma' \vdash a, c : A, C \dashv \Delta'}{\Gamma; \Gamma' \vdash \left[f\ a\,^{\mathrm{c}}\right], (v\, c) : B, C \dashv \Delta; \Delta'}\ \mathrm{M}_c$$

$$\frac{\Gamma \vdash f, v : A \multimap_{\mathrm{a}} B, C \multimap D \dashv \Delta \quad \Gamma' \vdash a, c : A, C \dashv \Delta'}{\Gamma; \Gamma' \vdash \left[f\ a\,^{\mathrm{a}}\right], (v\, c) : B, C \dashv \Delta; \Delta'}\ \mathrm{M}_c$$

$$\frac{\Gamma \vdash a, d : A^C_B, D^F_E \dashv \Delta \quad t, x : A, D; \Gamma' \vdash b, e : B, E \dashv \Delta'}{\Gamma; \Gamma' \vdash a_t\, b, d_x\, e : C, F \dashv \Delta; \Delta'}\ \mathrm{G}$$

$$\frac{\Gamma \vdash a, b : A, B^D_C \dashv \Delta}{\Gamma \vdash a, x : A, B \dashv b_x : B^D_C; \Delta}\ \mathrm{C}\ (x \text{ fresh}) \qquad \frac{\vdash e, c : E, C \dashv b_x : B^D_C; \Delta}{\Gamma \vdash e, (b_x\, c) : E, D \dashv \Delta}\ \mathrm{R}$$

Example of a simple interface derivation:

$$\dfrac{\dfrac{\vdots\,\pi}{\vdash \left[\mathsf{liked\ Sandy}^{\,c}\right], \mathsf{like'\ Sandy'} : np \multimap_{\mathrm{s}} s, \iota \multimap \pi \dashv} \quad \dfrac{}{\vdash \mathsf{Chris}, \mathsf{Chris} : np, \iota \dashv}\ \mathrm{Lex}}{\vdash \left[{}^{s}\mathsf{Chris}\ \left[\mathsf{liked\ Sandy}^{\,c}\right]\right], \mathsf{like'\ Sandy'\ Chris'} : s, \pi \dashv}\ \mathrm{M}_s$$

$$\pi = \dfrac{\dfrac{}{\vdash \mathsf{liked}, \mathsf{like'} : np \multimap_{\mathrm{c}} np \multimap_{\mathrm{s}} s, \iota \multimap \iota \multimap \pi \dashv}\ \mathrm{Lex} \quad \dfrac{}{\vdash \mathsf{Sandy}, \mathsf{Sandy'} : np, \iota \dashv}\ \mathrm{Lex}}{\vdash \left[\mathsf{liked\ Sandy}^{\,c}\right], \mathsf{like'\ Sandy'} : np \multimap_{\mathrm{s}} s, \iota \multimap \pi \dashv}\ \mathrm{M}_c$$

Example using the G rule

$$\dfrac{\vdots\,\pi_1 \qquad\qquad \vdots\,\pi_2}{\vdash \left[\mathsf{Sandy\ top}^{\,a}\right], \mathsf{top'\ Sandy'} : np^{t}_{s}, \iota^{\pi}_{\pi} \dashv \quad t, x : np, \iota \vdash \left[{}^{s}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,c}\right]\right], \mathsf{like'}\ x\ \mathsf{Chris'} : s, \pi \dashv}{\vdash \left[\mathsf{Sandy\ top}^{\,a}\right](\lambda t.\left[{}^{s}\mathsf{Chris}\ \left[\mathsf{liked}\ t^{\,c}\right]\right]), (\mathsf{top'\ Sandy'})(\lambda x.\mathsf{like'}\ x\ \mathsf{Chris'}) : t, \pi \dashv}\ \mathrm{G}$$

with trivial derivations for π_1 and π_2.

B On CVG Derivations

Proposition 1. *Let π be a CVG semantic derivation. It can be turned into a CVG semantic derivation where all C and R pairs of rule have been replaced by the above schema, and which derives the same term.*

Proof. This is proved by induction on the derivations. If the derivation stops on a Lexicon, Trace, Modus Ponens, G or C rule, this is trivial by application of the induction hypothesis.

If the derivation stops on a R rule, the C and R pair has the above schema. Note that nothing can be erased from Γ in π_2 because every variable in Γ occur (freely) only in a and Δ. So using a G rule (the only one that can delete material from the left hand side of the sequent) would leave variables in the store that could not be bound later. The same kind of argument shows that nothing can be retrieved from Δ before a_x had been retrieved. This means that no R rule can occur in π_2 whose corresponding C rule is in π_1 (while there can be a R rule with a corresponding C rule introduced in π_2). Hence we can make the transform and apply the induction hypothesis to the two premises of the new G rule.

C How to Build an Applicative ACG

Let $\Sigma_{\mathrm{HO}} = \langle A_{\mathrm{HO}}, C_{\mathrm{HO}}, \tau_{\mathrm{HO}} \rangle$. This section section shows how to build an ACG $\mathscr{G} = \langle \Sigma_{\mathrm{2nd}}, \Sigma_{\mathrm{HO}}, \mathscr{L}, s' \rangle$ such that $\mathcal{O}(\mathscr{G})$ is the set of $t : s \in \Lambda_{\Sigma_{\mathrm{HO}}}$ such that there exists π a proof of $\vdash_{\Sigma_{\mathrm{HO}}} t : s$ and π does not use the abstraction rule. This construction is very similar to the one given in [16, Chap. 7].

Definition 2. *Let α be a type. We inductively define the set* Decompose(α) *as:*

- *if α is atomic,* Decompose$(\alpha) = \{\alpha\}$*;*
- *if $\alpha = \alpha_1 \multimap \alpha_2$,* Decompose$(\alpha) = \{\alpha\} \cup \{\alpha_1\} \cup$ Decompose(α_2).

Let T be a set of types. We then define:

- $\mathsf{Base}(T) = \cup_{\alpha \in T} \mathsf{Decompose}(T)$;
- $\mathsf{At}(T)$ *a set of fresh atomic types that is in a one to one correspondence with* $\mathsf{Base}(T)$. *We note* $:=$ *one of the correspondence from* $\mathsf{At}(T)$ *to* $\mathsf{Base}(t)$ *(we also note* $:=$ *its unique homomorphic extension that is compatible with* $\multimap$. *The later is not necessarily a bijection);*
- *let* $\alpha \in \mathsf{Base}(T)$. *The set* $\mathsf{AtP}_T(\alpha)$ *of its* atomic profiles *is inductively defined as:*
  - *if* α *is atomic,* $\mathsf{AtP}_T(\alpha) = \{\alpha'\}$ *such that* α' *is the unique element of* $\mathsf{At}(T)$ *and* $\alpha' := \alpha$;
  - *if* $\alpha = \alpha_1 \multimap \alpha_2$, $\mathsf{AtP}_T(\alpha) = \{\alpha'\} \cup \{\alpha'_1 \multimap \alpha'_2 \mid \alpha'_2 \in \mathsf{AtP}_T(\alpha_2)\}$ *where:*
    - α' *is uniquely defined in* $\mathsf{At}(T)$ *and* $\alpha' := \alpha$;
    - α'_1 *is uniquely defined in* $\mathsf{At}(T)$ *and* $\alpha'_1 := \alpha_1$. *There exists such an* α'_1 *because* $\alpha_1 \in \mathsf{Decompose}(\alpha)$ *and* $\mathsf{Decompose}(\alpha) \subset \mathsf{Base}(T)$ *when* $\alpha \in \mathsf{Base}(T)$.

    Note that for the same reason, α'_2 *is well defined.*

Note that for any $\alpha \in \mathsf{Base}(T)$, The types in $\mathsf{AtP}_T(\alpha)$ are of order at most 2.

Proposition 2. *Let* T *be a set of types and* $\alpha \in \mathsf{Base}(T)$ *with* $\alpha = \alpha_1 \multimap \ldots \multimap \alpha_k \multimap \alpha_0$ *such that* α_0 *is atomic. Then* $|\mathsf{AtP}_T(\alpha)| = k + 1$.

Proof. By induction.

Proposition 3. *Let* T *be a set of types and* $\alpha \in \mathsf{Base}(T)$. *Then for all* $\alpha' \in \mathsf{AtP}_T(\alpha)$ *we have* $\alpha' := \alpha$.

Proof. By induction.

In the following, we always consider $T = \cup_{c \in C_{\mathrm{HO}}} \tau_{\mathrm{HO}}(c)$. We then can define $\Sigma_{2\mathrm{nd}} = \langle A_{2\mathrm{nd}}, C_{2\mathrm{nd}}, \tau_{2\mathrm{nd}} \rangle$ with:

- $A_{2\mathrm{nd}} = \mathsf{At}(T)$
- $s' \in A_{2\mathrm{nd}}$ the unique term such that $s' := s$
- $C_{2\mathrm{nd}} = \cup_{c \in C_{\mathrm{HO}}} \{\langle c, \alpha' \rangle | \alpha' \in \mathsf{AtP}_T(\tau_{\mathrm{HO}}(c))\}$ ($\mathsf{AtP}_T(\tau_{\mathrm{HO}}(c))$ is well defined because $\tau_{\mathrm{HO}}(c) \in \mathsf{Base}(T)$)
- for every $c' = \langle c, \alpha' \rangle \in C_{2\mathrm{nd}}$, $\tau_{2\mathrm{nd}}(c') = \alpha'$

Note that according to Proposition 2, for every constant c of C_{HO} of arity k (i.e. $\tau_{\mathrm{HO}}(c) = \alpha_1 \multimap \ldots \multimap \alpha_k \multimap \alpha_0$), there are $k + 1$ constants in $C_{2\mathrm{nd}}$.

Finally, in order to completely define $\mathscr{G}$, we need to define $\mathscr{L}$:

- for $\alpha' \in A_{2\mathrm{nd}}$, there exists a unique $\alpha \in \mathsf{Base}(T)$ such that $\alpha' := \alpha$ by construction of $\mathsf{At}(T)$. We set $\mathscr{L}(\alpha') = \alpha$.
- for $c' = \langle c, \alpha' \rangle \in C_{2\mathrm{nd}}$, we set $\mathscr{L}(c') = c$

According to Proposition 3, we have $\mathscr{L}(\tau_{2\mathrm{nd}}(c')) = \alpha$ where α is the type of $\mathscr{L}(c')$ so $\mathscr{L}$ is well defined.

Proposition 4. *There exists $t : \alpha \in \Lambda_{\Sigma_{\mathrm{HO}}}$ build using only applications if and only if there exists $t' : \alpha'$ a closed term of $\Lambda_{\Sigma_{2\mathrm{nd}}}$ with α' the unique element of $\mathsf{At}(T)$ such that $\alpha' := \alpha$ and $\mathscr{L}(t') = t$.*

Proof. $\Rightarrow$ We prove it by induction on t. If t is a constant, we take $t' = \langle t, \alpha'$ with α' the unique element of $\mathsf{At}(T)$ such that $\alpha' := \alpha$. By definition, $\mathscr{L}(t') = t$. If $t = c\, u_1 \dots u_k$, then $c \in C_{\mathrm{HO}}$ is of type $\alpha_1 \multimap \dots \multimap \alpha_k \multimap \alpha$ and for all $i \in [1, k]$ u_k is of type α_i. We know there exist $c' = \langle c, \beta' \rangle \in \Sigma_{2\mathrm{nd}}$ such that $\beta' = \alpha'_1 \multimap \dots \alpha'_k \multimap \alpha'$ with for all $i \in [1, k]$, α'_i is the unique element of $\mathsf{At}(T)$ such that $\alpha'_i := \alpha_i$ and α' the unique element of $\mathsf{At}(T)$ such that $\alpha' := \alpha$. By induction hypothesis, we also have for all $i \in [1, k]$ a term $u'_i : \alpha'_i$ with α'_i the unique element of $\mathsf{At}(T)$ such that $\alpha'_i := \alpha_i$ and $\mathscr{L}(u'_i) = u_i$. If we take $t' = \langle c, \beta' \rangle\, u'_1 \dots u'_k$, we have $\mathscr{L}(t') = \mathscr{L}(\langle c, \beta' \rangle\, u'_1 \dots u'_k) = \mathscr{L}(\langle c, \beta' \rangle)\, \mathscr{L}(u'_1) \dots \mathscr{L}(u'_k) = c\, u_1 \dots u_k = t$ which completes the proof.

$\Leftarrow$ If $\alpha' \in \mathsf{At}(T)$ and t' is a closed term then because $\Sigma_{2\mathrm{nd}}$ is of order 2, then t' is build only using applications. Hence its image by $\mathscr{L}$ is also only build using applications.

Observational Effort and Formally Open Mappings

Bernhard Heinemann

Fakultät für Mathematik und Informatik,
FernUniversität in Hagen,
58084 Hagen, Germany
bernhard.heinemann@fernuni-hagen.de

Abstract. Starting off with Moss and Parikh's investigation into knowledge and topology, we propose a logical system which is capable of formally handling endomorphisms of subset spaces. The motivation for doing so originates from dynamic agent logics. Usually, these logics comprise certain epistemic actions. Our aim is to show that an appropriate extension of the Moss-Parikh system can serve similar purposes. In fact, since the semantics of an action can be described as a function inducing a change of the knowledge states of the involved agents, such transformations are to be modeled accordingly. Due to the ambivalence of the framework used here, this has some quasi-topological impact, too, in so far as a certain notion of open mapping can be captured now. The main issues of this paper concern the basic logical properties of the arising system, in particular, completeness. Our main technical resource for that is hybrid logic.

Keywords: topology and epistemic logic, subset spaces, hybrid logic, open mappings.

1 Introduction

Caused by the demands for spatial reasoning in AI, a renewed interest in an old topic has recently got a good response in the relevant literature: the modal logic view of topology, initiated by Tarski approximately 70 years ago; cf [1] or [2]. A comprehensive overview of the modern developments referring to this can be found in the handbook [3].

Different priorities have been set for analysing the relationship between topology and modal logic. We here focus on a particular approach due to Moss and Parikh which not only brings about a modern account of the classical results of McKinsey and Tarski, but also stresses the information-theoretic content of topological spaces as epistemic structures; cf [4] or [5]. The present paper is based on this work of Moss and Parikh so that the essentials of their system should be mentioned in advance. Basically, the underlying language, $\mathcal{L}$, contains a modality K describing the *knowledge* of an agent under discussion, and a second one, $\Box$, formalizing *observational effort* to acquire knowledge. The semantic domains are

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 197–208, 2009.

triples $(S, \mathcal{O}, V)$ called *subset spaces*. These models consist of a non-empty set S of states, a set $\mathcal{O}$ of subsets of S representing the knowledge states of the agent,[1] and a valuation V determining the states where the atomic propositions are true. The $\mathcal{L}$-formulas are interpreted in subset spaces with respect to *neigbourhood situations* x, U, where $x \in U \in \mathcal{O}$. In particular, the operator K quantifies over all states that are taken from some fixed knowledge state $U \in \mathcal{O}$, whereas $\Box$ quantifies 'downward' over all $U' \in \mathcal{O}$ that are contained in U since shrinking and gaining knowledge correspond to each other. Thus, compared to the original framework the class of admissible domains is indeed expanded, but topological spaces can be characterised

The generality of $\mathcal{L}$ is accompanied by its weak expressiveness. With regard to special purpose applications it is, therefore, desirable to strengthen the language appropriately. To this end, certain *functions* were added to $\mathcal{L}$ in the papers [6] and [7], respectively. In epistemic contexts, functions play an important part since they appear, eg, as the semantics of actions of agents. That is why integrating functions constitutes some kind of *dynamic epistemic logic,* in the spirit of [8].

It turned out that additional means of expression were needed for doing this successfully. In fact, *hybrid logic,* cf [9], Sect. 7.3, provides a suitable basis for dealing with functions in connection with subset spaces. In the paper [10], a basic logical system, called HS, was developed, which contains nominals for both states and sets, and the global modality, cf [9], Sect. 7.1, for simulating hybrid satisfaction operators.

The system HS is structurally rather simple thus. But it gets it about right, apart from one crucial point: It is impossible that a state-valued function f is forced to transform the actual neighbourhood onto an element of $\mathcal{O}$, or, in other words, to respect knowledge states. The latter exactly means that f is an endomorphism of subset spaces, or – as the reader will see when we give the precise definitions –, an "open mapping". However, the ability to formally specify such an important property is very desirable, especially in view of dynamic epistemic logic. Hence that deficiency will be rectified in the present paper.

For this purpose, the system HS must be upgraded first. We now take also the hybrid $\downarrow$-binder, cf [11], Sect. 2, into account, and we design a correspondingly enriched hybrid logic of subset spaces. This makes up the bulk of the paper, before we will be able to conclude our actual intention.

In detail, the technical part of the paper is organised as follows. In Section 2, we define and discuss the language our study is based on. To be more precise, we first fix the function-free part of the language and incorporate functions into it afterwards. The next section then deals with the main model-theoretic aspect of the arising logic, viz completeness. Subsequently, in Section 4, we show that our sytem is strong enough to treat open mappings logically. The paper is concluded with a summary and a discussion of the achievements we have obtained so far, some hints to future research and a comparison with other approaches.

[1] Due to the topological connection, the elements of $\mathcal{O}$ are sometimes called the *opens* here. This is in accordance with the common manner of speaking, eg, in the fundamental paper [4].

2 The Upgraded Hybrid Language

We now extend the hybrid language for subset spaces from [10]. The new language should additionally contain the $\downarrow$-binder. In this case, it is common to have also *state variables* at one's disposal; cf [11], loc cit. Since the source language is *sorted,* we will have two sorts of variables here though, in fact state variables and *set variables.*

Let be given five mutually disjoint denumerably-infinite sets of symbols, $\mathrm{Prop} = \{p, q, \ldots\}$, $\mathrm{N}_{stat} = \{i, j, \ldots\}$, $\mathrm{V}_{stat} = \{x, y, \ldots\}$, $\mathrm{N}_{sets} = \{I, J, \ldots\}$ and $\mathrm{V}_{sets} = \{X, Y, \ldots\}$. The elements of these sets are called *proposition variables, names of states, state variables, names of sets* and *set variables,* respectively. The set Form of *formulas* over $\mathrm{Prop} \cup \mathrm{N}_{stat} \cup \mathrm{V}_{stat} \cup \mathrm{N}_{sets} \cup \mathrm{V}_{sets}$ is then defined by the rule

$$\alpha \ ::= \ p \mid i \mid x \mid I \mid X \mid \neg\alpha \mid \alpha \wedge \alpha \mid \mathsf{K}\alpha \mid \Box\alpha \mid \mathsf{A}\alpha \mid \downarrow x.\alpha \mid \downarrow X.\alpha.$$

The missing boolean connectives $\top, \bot, \vee, \rightarrow, \leftrightarrow$ are treated as abbreviations, as needed. The duals of the modal operators K, $\Box$ and A are denoted L, $\Diamond$ and E, respectively. The $\downarrow$-binder turns out to be self-dual.

The semantic domains of our language are subset spaces where the usual modal valuations are extended to nominals. Moreover, *assignments to variables* have to be taken into account as well. All this is contained in the following definition, in which the powerset of a given set S is designated $\mathcal{P}(S)$.

Definition 1 (Subset frames; hybrid subset spaces; assignments).

1. *A* subset frame *is a pair $\mathcal{S} = (S, \mathcal{O})$, where S is a non-empty set (of* states*) and $\mathcal{O} \subseteq \mathcal{P}(S)$ a set of subsets of S.*

Now, let $\mathcal{S} = (S, \mathcal{O})$ be a subset frame.

2. *The set $\mathcal{N}_{\mathcal{S}} := \{s, U \mid s \in U \text{ and } U \in \mathcal{O}\}$ is called the* set of neighbourhood situations *of $\mathcal{S}$. (Although neighbourhood situations are pairs, they are mostly written without brackets.)*
3. *An $\mathcal{S}$-*valuation *is a mapping $V : \mathrm{Prop} \cup \mathrm{N}_{stat} \cup \mathrm{N}_{sets} \to \mathcal{P}(S)$ such that*
   (a) $V(p) \in \mathcal{P}(S)$ for all $p \in \mathrm{Prop}$,
   (b) $V(i)$ either equals $\emptyset$ or is a singleton subset of S for every $i \in \mathrm{N}_{stat}$, and
   (c) $V(I) \in \mathcal{O}$ for every $I \in \mathrm{N}_{sets}$.
4. *A triple $\mathcal{M} := (S, \mathcal{O}, V)$, where $\mathcal{S} = (S, \mathcal{O})$ is a subset frame and V an $\mathcal{S}$-valuation, is called a* hybrid subset space *(or, in short, an* HSS*). We say that $\mathcal{M}$ is* based on *$\mathcal{S}$ then.*
5. *An $\mathcal{S}$-*assignment[2] *is a mapping $g : \mathrm{V}_{stat} \cup \mathrm{V}_{sets} \to \mathcal{P}(S)$ such that*
   (a) $g(x)$ is either $\emptyset$ or a singleton subset of S for every $x \in \mathrm{V}_{stat}$, and
   (b) $g(X) \in \mathcal{O}$ for every $X \in \mathrm{V}_{sets}$.

[2] If the underlying frame is clear, then the prefix '$\mathcal{S}$' is sometimes omitted. The same applies for valuations.

Note that it is generally accepted for the original language $\mathcal{L}$ that $\{S, \emptyset\} \subseteq \mathcal{O}$; cf [4], Sect. 1.1. This is not required from the outset here. But note that nominals and variables, respectively, may have an empty denotation nevertheless, which is appropriate to us for technical reasons, but is not common in standard hybrid logic.

Neighbourhood situations as well as assignments are used for defining the relation of satisfaction being relevant to hybrid subset spaces $\mathcal{M} = (S, \mathcal{O}, V)$. It is convenient to introduce a little more notation in advance: Let $\xi \in \mathrm{V}_{stat} \cup \mathrm{V}_{sets}$, and let $\Phi \in S \cup \mathcal{O}$ be of the same type as ξ. Furthermore, let g be any assignment. Then g_ξ^Φ denotes the assignment which equals g except for the argument ξ which is assigned to Φ.

Given an HSS $\mathcal{M}$ based on a frame $\mathcal{S}$ and an $\mathcal{S}$-assignment g, we now define the relation of satisfaction, $\models_{\mathcal{M},g}$, between the neighbourhood situations of $\mathcal{S}$ and the formulas from Form.

Definition 2 (Satisfaction and validity).

1. *Let $\mathcal{M} = (S, \mathcal{O}, V)$ be an HSS based on $\mathcal{S} = (S, \mathcal{O})$, g an $\mathcal{S}$-assignment and s, U a neighbourhood situation of $\mathcal{S}$. Then*

$$
\begin{array}{lll}
s,U \models_{\mathcal{M},g} p & :\Longleftrightarrow & s \in V(p) \\
s,U \models_{\mathcal{M},g} i & :\Longleftrightarrow & s \in V(i) \\
s,U \models_{\mathcal{M},g} x & :\Longleftrightarrow & s \in g(x) \\
s,U \models_{\mathcal{M},g} I & :\Longleftrightarrow & V(I) = U \\
s,U \models_{\mathcal{M},g} X & :\Longleftrightarrow & g(X) = U \\
s,U \models_{\mathcal{M},g} \neg\alpha & :\Longleftrightarrow & s,U \not\models_{\mathcal{M},g} \alpha \\
s,U \models_{\mathcal{M},g} \alpha \wedge \beta & :\Longleftrightarrow & s,U \models_{\mathcal{M},g} \alpha \text{ and } s,U \models_{\mathcal{M},g} \beta \\
s,U \models_{\mathcal{M},g} \mathsf{K}\alpha & :\Longleftrightarrow & t,U \models_{\mathcal{M},g} \alpha \text{ for all } t \in U \\
s,U \models_{\mathcal{M},g} \Box\alpha & :\Longleftrightarrow & \forall U' \in \mathcal{O} : (s \in U' \subseteq U \Rightarrow s,U' \models_{\mathcal{M},g} \alpha) \\
s,U \models_{\mathcal{M},g} \mathsf{A}\alpha & :\Longleftrightarrow & t,U' \models_{\mathcal{M},g} \alpha \text{ for all } t,U' \in \mathcal{N}_{\mathcal{S}} \\
s,U \models_{\mathcal{M},g} \downarrow x.\alpha & :\Longleftrightarrow & s,U \models_{\mathcal{M},g_x^s} \alpha \\
s,U \models_{\mathcal{M},g} \downarrow X.\alpha & :\Longleftrightarrow & s,U \models_{\mathcal{M},g_X^U} \alpha,
\end{array}
$$

   where $p \in \mathrm{Prop}$, $i \in \mathrm{N}_{stat}$, $x \in \mathrm{V}_{stat}$, $I \in \mathrm{N}_{sets}$, $X \in \mathrm{V}_{sets}$, and $\alpha, \beta \in \mathrm{Form}$. In case $s,U \models_{\mathcal{M},g} \alpha$ is true we say that α holds in $\mathcal{M}$ at *the neighbourhood situation s,U under the assignment g.*
2. *Let $\mathcal{M}$ be an HSS and g an assignment. A formula α is called* valid in $\mathcal{M}$ under g, *iff it holds in $\mathcal{M}$ under g at every neighbourhood situation s,U of the subset frame $\mathcal{M}$ is based on. (Manner of writing: $\mathcal{M}, g \models \alpha$.)*

Let $\varphi \in \mathrm{N}_{stat} \cup \mathrm{V}_{stat}$ and $\Psi \in \mathrm{N}_{sets} \cup \mathrm{V}_{sets}$. Then, for a fixed assignment, the formula $\varphi \wedge \Psi$ holds at a single neighbourhood situation at most. Thus it is possible to identify neighbourhood situations by formulas of this type. Since neighbourhood situations constitute the semantic atoms of the language of subset frames, it therefore makes sense to associate a *satisfaction operator* with such a formula.

By the following definition, satisfaction operators are properly integrated into the hybrid language for subset spaces:

$$@_{(\varphi\wedge\Psi)}\alpha :\equiv \mathsf{E}(\varphi \wedge \Psi \wedge \alpha),$$

for all $\alpha \in$ Form.

For the rest of this section, subset frames *with funtions* are underlain our treatment. For simplicity, we add only one function, F, which is represented by a further modal operator, $[F]$. We retain the designation Form for the arising set of formulas.

Definition 3 (Functional subset frames). *Let $(S, \mathcal{O})$ be a subset frame and $F : S \to S$ a function. Then the triple $\mathcal{S} := (S, \mathcal{O}, F)$ is called a* functional subset frame. *The modifications of the other notions from Definition 1 are derived from this in a straightforward manner.*

As it has already been indicated in the introduction, we are particularly interested in subset frames with endomorphisms.

Definition 4 (Preserving opens). *A functional subset frame $\mathcal{S} = (S, \mathcal{O}, F)$ is said to* preserve opens, *iff $F(U) \in \mathcal{O}$ holds for all $U \in \mathcal{O}$. In this case, the mapping F is called* formally open.

We now extend the notion of satisfaction to functional hybrid subset spaces preserving opens. For convenience, we call these structures simply the *pertinent HSSs*.

Definition 5 (Satisfaction). *Let $\mathcal{M} = (S, \mathcal{O}, F, V)$ be a pertinent HSS based on $\mathcal{S} = (S, \mathcal{O}, F)$, g an $\mathcal{S}$-assignment and s, U a neighbourhood situation of $\mathcal{S}$. Then*

$$s, U \models_{\mathcal{M},g} [F]\alpha :\iff F(s), F(U) \models_{\mathcal{M},g} \alpha,$$

where $\alpha \in$ Form. The notion of validity is defined as in Definition 2.

With that, the new language is completely specified. – The following proposition, saying that the operator $[F]$ is self-dual, is obvious.

Proposition 1 (Self-duality). *Let $\mathcal{M} = (S, \mathcal{O}, F, V)$ be a pertinent HSS based on $\mathcal{S} = (S, \mathcal{O}, F)$, g an $\mathcal{S}$-assignment and s, U a neighbourhood situation of $\mathcal{S}$. Then we have that*

$$s, U \models_{\mathcal{M},g} \neg[F]\alpha \;\; \textit{iff} \;\; s, U \models_{\mathcal{M},g} [F]\neg\alpha.$$

The next proposition gives an example of the expressiveness of the new language. We establish a particular validity which will play an important part in the course of this paper since it forces formal openness.

Proposition 2. *Let $\mathcal{M}$ and g be as in the previous proposition. Then,*

$$\mathcal{M}, g \models \downarrow x.\, \downarrow X.\, [F]\mathsf{K} \downarrow y.\, @_{(x\wedge X)}\mathsf{L}[F]y.$$

The proof of this proposition relies on Definition 2 and Definition 5. It can be found in the appendix.

3 Hybrid Completeness

First in this section, we give a brief review of those features of HS which are also significant to the logic resulting from the full language. We then deal with the necessary extensions.

Concerning *axioms,* the reader is referred to Sect. 3 of the paper [10]. For us, only the *properties* these axioms imply are important since we want to outline the model construction underlying the corresponding completeness proof. Our starting point is a maximal consistent set Γ of formulas containing a given HS-consistent formula γ. By means of an appropriate *Lindenbaum Lemma,* Γ can be extended to a certain maximal consistent set Γ' of formulas in a language $\mathcal{L}'$ containing enough nominals. It is one of the particular properties of Γ' that this set is *named,* i.e., Γ' contains some state name and some set name. Let $\xrightarrow{\Delta}$ be the accessibility relation belonging to the modality $\Delta \in \{\mathsf{K}, \Box, \mathsf{A}\}$ of the canonical model of HS relative to $\mathcal{L}'$.[3] Then, the second important property of Γ' ensures that *named witnesses are provided,* i.e., the submodel $\mathcal{M}'$ having domain $D := \{\Sigma \mid \Sigma \text{ is named and } \Gamma' \xrightarrow{\mathsf{A}} \Sigma\}$ of that canonical model satisfies the *Existence Lemma;* cf [10], 3.3 and 3.6.

Since the operator K follows the S5 laws, the relation $\xrightarrow{\mathsf{K}}$ is an equivalence. Let $[\Sigma] := \{\Sigma' \in D \mid \Sigma \xrightarrow{\mathsf{K}} \Sigma'\}$ denote the $\xrightarrow{\mathsf{K}}$-equivalence class of the point $\Sigma \in D$, and let $\mathcal{Q} := \{[\Sigma] \mid \Sigma \in D\}$ be the set of all such classes. The following relation of precedence between elements of $\mathcal{Q}$ proves to be useful:

$$[\Sigma] \preccurlyeq [\Theta] : \Longleftrightarrow \exists\, \Sigma' \in [\Sigma], \Theta' \in [\Theta] : \Sigma' \xrightarrow{\Box} \Theta',$$

for all $\Sigma, \Theta \in D$. It turns out that $\preccurlyeq$ is a left-directed partial order. Moreover, if $[\Sigma] \preccurlyeq [\Theta]$ is true, then the relation $\xrightarrow{\Box}$ restricted to $[\Sigma]$ in the domain and $[\Theta]$ in the range is an injective and surjective partial function; see [10], 3.8. Now, for all $[\Sigma], [\Theta] \in \mathcal{Q}$ such that $[\Sigma] \preccurlyeq [\Theta]$, let $f^{[\Theta]}_{[\Sigma]} : [\Theta] \to [\Sigma]$ be the mapping which is inverse to that function. Let S be the set of all partial functions $f : \mathcal{Q} \to D$ of which the domain $\mathrm{dom}(f)$ is a maximal subset of $\mathcal{Q}$ regarding the following two conditions:

1. $f([\Sigma]) \in [\Sigma]$ for all $[\Sigma] \in \mathrm{dom}(f)$,
2. $f([\Sigma]) = f^{[\Theta]}_{[\Sigma]}(f([\Theta]))$ for all $[\Sigma], [\Theta] \in \mathrm{dom}(f)$ such that $[\Sigma] \preccurlyeq [\Theta]$.

We write $f_\Sigma := f([\Sigma])$ if $f([\Sigma])$ exists. The set S serves as the carrier set of the desired model $\mathcal{M}$. We also recall the definition of the distinguished set of subsets. Let $U_{[\Sigma]} := \{f \in S \mid f_\Sigma \text{ exists}\}$, for all $\Sigma \in D$; then let $\mathcal{O} := \{U_{[\Sigma]} \mid \Sigma \in D\} \cup \{\emptyset\}$. The valuation V of $\mathcal{M}$ is derived from the canonical one in a standard manner.

It is worth mentioning that a function $f \in S$ is already determined by its value for a single argument; see [10], 3.9. On the other hand, every $\Sigma \in D$ actually

[3] Later on, we denote the accessibility relations for further modalities correspondingly.

induces a function passing through this point, according to the definition of S. Hence this function is unique and denoted f^{Σ} thus.

Finally, a few remarks on the nominal structure of $\mathcal{M}'$ and $\mathcal{M}$, respectively. First, it should be mentioned that not only every point of $\mathcal{M}'$ is named, but also that every pair i, I, where $i \in \mathrm{N}_{stat}$ and $I \in \mathrm{N}_{sets}$, is the 'name' of at most one such point. In other words, the formulas $i \wedge I$ act as proper nominals for $\mathcal{M}'$; cf the remark right after Definition 2 and see [10], 3.5.3. Second, a state name $i \in \mathrm{N}_{stat}$ is constant along every function $f \in S$ as follows immediately from one of the Axioms of HS ([10], Sect. 3, Axiom 11). And third, we have the following strong uniqueness property for names of states with regard to $\mathcal{M}$. If $i \in \mathrm{N}_{stat}$ is contained in $\Sigma \cap \Sigma'$, where $\Sigma, \Sigma' \in D$, then $f^{\Sigma} = f^{\Sigma'}$. This fact can easily be concluded from the previous one, the left-directedness of $\preccurlyeq$ and [10], 3.5.5.

The subsequent *Truth Lemma* was proved in [10], 3.11, from which the completeness of HS with respect to the class of all HSSs follows readily.

Lemma 1. *The model $\mathcal{M}$ is an HSS such that for all formulas α, functions $f \in S$, and points $\Sigma \in D$ satisfying $f \in U_{[\Sigma]}$, we have that α holds in $\mathcal{M}$ at the neighbourhood situation $f, U_{[\Sigma]}$ iff $\alpha \in f_{\Sigma}$.*

This completes the review of HS. We now argue that the same way of proceeding can be utilised for the extended system, ad hoc called EHS, as well. For a start, the treatment of variables is completely analogous to that of nominals. We have, in particular, the schemata 11 – 14 from [10] also for state and set variables, respectively. But variables need not be included in the naming process connected to the Lindenbaum Lemma. Hence it is clear how the assignment g associated with $\mathcal{M}$ has to be defined. Furthermore, we can directly go on with the $\downarrow$-case of the Truth Lemma. Two additional axioms are needed for handling it successfully, which read

$$(1)\ \mathsf{A}\,(i \to (\downarrow x.\, \alpha \leftrightarrow \alpha[x/i])) \quad \text{and} \quad (2)\ \mathsf{A}\,(I \to (\downarrow X.\, \alpha \leftrightarrow \alpha[X/I]))\,,$$

where $i \in \mathrm{N}_{stat}$, $I \in \mathrm{N}_{sets}$, $x \in \mathrm{V}_{stat}$, $X \in \mathrm{N}_{sets}$ and $\alpha \in \mathrm{Form}$; substitution (of variables with nominals) is here defined as in first-order logic, where binding only concerns the operator $\downarrow$. As one will see in a minute, these axioms effect a reduction of the $\downarrow$-case to the nominal case so that one can exploit the nominal structure of $\mathcal{M}$ appropriately.

Lemma 2 (Truth Lemma). *Let $\mathcal{M}$ and g be as above. Then, for all $\alpha \in \mathrm{Form}$, $f \in S$, and $\Sigma \in D$ satisfying $f \in U_{[\Sigma]}$, we have that $f, U_{[\Sigma]} \models_{\mathcal{M},g} \alpha$ iff $\alpha \in f_{\Sigma}$.*

Proof. Only the case of the $\downarrow$-binder has to be considered. We confine ourselves to state variables since the case of set variables is similar. So let $\alpha = \downarrow y.\, \beta$. Moreover, Let f and Σ be given such that $f \in U_{[\Sigma]}$. Take a nominal $i \in f_{\Sigma}$. (Such a nominal actually exists.) Due to the construction of $\mathcal{M}$, we then have $V(i) = f$. Consequently, we get

$$
\begin{array}{lll}
f, U_{[\Sigma]} \models_{\mathcal{M},g} \downarrow y.\, \beta & \Longleftrightarrow & f, U_{[\Sigma]} \models_{\mathcal{M},g^f_y} \beta \quad \text{(by Definition 2)} \\
& \Longleftrightarrow & V(i), U_{[\Sigma]} \models_{\mathcal{M},g^{V(i)}_y} \beta \quad \text{(since } V(i) = f\text{)} \\
& \Longleftrightarrow & V(i), U_{[\Sigma]} \models_{\mathcal{M},g} \beta[y/i] \quad \text{(by induction on } \beta\text{)} \\
& \Longleftrightarrow & \beta[y/i] \in f_\Sigma \quad \text{(by induction hypothesis)} \\
& \Longleftrightarrow & \downarrow y.\, \beta \in f_\Sigma \quad \text{(due to Axiom (1)).}
\end{array}
$$

This proves the lemma in the case we selected in advance.

As an immediate consequence of Lemma 2 we obtain the desired completeness result for EHS.

Theorem 1 (Hybrid completeness). *The system* EHS *is sound and complete with respect to the class of all HSSs.*

4 The Logic of Pertinent HSSs

In the main part of this paper following now, we give an axiomatic description of pertinent hybrid subset spaces. Focussing on the operator $[F]$ we prove a corresponding completeness theorem. Some comments on the upcoming logic are postponed to the concluding section. – The validities concerning $[F]$ read as follows.

1. $[F](\alpha \to \beta) \to ([F]\alpha \to [F]\beta)$
2. $\neg[F]\alpha \leftrightarrow [F]\neg\alpha$
3. $\Diamond[F]i \to \Box[F]i$
4. $\mathsf{K}[F]\,(\alpha \to \mathsf{L}\beta) \vee \mathsf{K}[F]\,(\beta \to \mathsf{L}\alpha)$
5. $\downarrow x.\, \downarrow X.\, [F]\mathsf{K} \downarrow y.\, @_{(x \wedge X)}\mathsf{L}[F]y$
6. $\mathsf{A}\alpha \to [F]\alpha$,

where $i \in \mathrm{N}_{stat}$, $x, y \in \mathrm{V}_{stat}$, $X \in \mathrm{V}_{sets}$ and $\alpha, \beta \in \mathrm{Form}$. The reader will easily check that this list in fact involves only validities. While this is obvious for (1) – (3) and (6), it is best seen indirectly in case of (4); as to (5), see Proposition 2.

By adding suitable proof rules for $[F]$[4] a logic called PHS results from these axioms. For PHS, we obtain the following theorem.

Theorem 2 (Extended completeness). PHS *is sound and complete with respect to the class of all pertinent HSSs.*

The rest of this section is devoted to the proof of Theorem 2. Keeping the notations from the previous section and omitting some routine arrangements, we define the function $F : S \to S$ interpreting the modality $[F]$ right away. Let f be an arbitrary element of S. Then there is some $\Theta \in D$ such that f_Θ is defined. Due to Axiom (2), there exists some $\Sigma \in D$ such that $f_\Theta \xrightarrow{[F]} \Sigma$. We now let

$$F(f) := f^\Sigma.$$

Clearly, we must show that F is correctly defined in this way.

[4] See [10], where corresponding rules are stated for the modalities of the basic language.

Lemma 3. *The mapping F is well-defined.*

Proof. It must be proved that the definition of F is independent of both Σ and Θ. So let us assume that $f_\Theta \xrightarrow{[F]} \Sigma$ as well as $f_{\Theta'} \xrightarrow{[F]} \Sigma'$ is valid. We know that there is some $i \in \mathrm{N}_{stat}$ contained in Σ. Using Axiom (2) among other things, it follows that $[F]i \in f_\Theta$. We now utilise that there exists some $\Xi \in D$ such that $f_\Xi \xrightarrow{\Box} f_\Theta$ and $f_\Xi \xrightarrow{\Box} f_{\Theta'}$. Thus $\Diamond[F]i \in f_\Xi$. With the aid of Axiom (3) we get $\Box[F]i \in f_\Xi$. Consequently, $[F]i \in f_{\Theta'}$, hence $i \in \Sigma'$. This implies $f^\Sigma = f^{\Sigma'}$, as it was stated in Sect. 3. Therefore, the definition of F is independent of both Σ and Θ, as desired.

Let the extension of the original model with F be equally designated $\mathcal{M}$. The decisive property of $\mathcal{M}$ is that opens are respected by the mapping F.

Lemma 4. *For all $U \in \mathcal{O}$, we have that $F(U) \in \mathcal{O}$.*

Proof. Let $U \in \mathcal{O}$ be given. Then there is some $\Sigma \in D$ such that $U = U_{[\Sigma]}$. Let Δ be the unique point of D satisfying $\Sigma \xrightarrow{[F]} \Delta$. We claim that $F(U) = U_{[\Delta]}$. To this end, the following two properties must be established.

(a) If $f \in U_{[\Sigma]}$, then $F(f) \in U_{[\Delta]}$.
(b) If $h \in U_{[\Delta]}$, then there exists $f \in U_{[\Sigma]}$ such that $h = F(f)$.

We first prove (a). For that, it suffices to show that if $f_\Sigma \xrightarrow{[F]} \Theta$, then $\Delta \xrightarrow{\mathsf{K}} \Theta$. Assuming the contrary gives us formulas $\alpha, \beta \in \mathrm{Form}$ such that

$$\alpha \in \Delta, \neg\mathsf{L}\alpha \in \Theta \text{ and } \beta \in \Theta, \neg\mathsf{L}\beta \in \Delta.$$

This implies $\mathsf{L}[F](\alpha \wedge \neg\mathsf{L}\beta) \wedge \mathsf{L}[F](\beta \wedge \neg\mathsf{L}\alpha) \in \Sigma$, contradicting Axiom (4). Thus the assumption was wrong, i.e., (a) is valid.

We now prove (b). This is the place where Axiom (5) comes into play. So let $h \in U_{[\Delta]}$ be given. Since every axiom is contained in Σ and every element of D is named, Axiom (5) and the two schemata from Sect. 3 imply that there is some $\Xi \in D$ such that $\Sigma \xrightarrow{\mathsf{K}} \Xi$ and $\Xi \xrightarrow{[F]} h_\Delta$; cf the proof of Proposition 2. This is what we wanted to show since $F\left(f^\Xi\right) = h$ is valid then according to the definition of F. – Now, $F(U) = U_{[\Delta]}$ follows readily from (a) and (b).

Consequently, the extended model $\mathcal{M}$ is a pertinent HSS. In order to complete the proof of Theorem 2 it remains to establish the Truth Lemma for the case involving $[F]$.

Lemma 5. *Lemma 2 holds for the system* PHS *as well.*

Proof. Let $\alpha = [F]\beta$. Moreover, Let f and Σ be given such that $f \in U_{[\Sigma]}$. Finally, let $h = F(f)$ and $F(U) = U_{[\Delta]}$. Then $h = f^\Theta$, where Θ is determined from $f_\Sigma \xrightarrow{[F]} \Theta$, and we get

$$\begin{aligned} f, U_{[\Sigma]} \models_{\mathcal{M},g} [F]\beta &\iff F(f), F\left(U_{[\Sigma]}\right) \models_{\mathcal{M},g} \beta \text{ (by Definition 5)} \\ &\iff \beta \in h_{\Delta} \qquad \text{(by induction hypothesis)} \\ &\iff [F]\beta \in f_{\Sigma} \qquad \text{(since } h_{\Delta} = \Theta\text{)}. \end{aligned}$$

This proves the lemma. – Altogether, the proof of Theorem 2 is completed with that.

5 Discussion

In the present paper, the basic hybrid logic for reasoning about knowledge and topology from [10] was extended with sorted variables for states and sets as well as with corresponding $\downarrow$-binders. In this way, the integration of endomorphisms into that system was facilitated. The final outcome was an axiomatic characterisation of the resulting logic by means of a corresponding soundness and completeness theorem.

Given a functional subset frame $\mathcal{S} = (S, \mathcal{O}, F)$ preserving opens, the mapping F need not be *contracting,* i.e., the property $F(U) \subseteq U$ need not necessarily be valid for all $U \in \mathcal{O}$. The fact that knowledge may get lost through certain epistemic actions accounts for that. However, if the contraction property is required, then we have the well-known *Cross Axiom* for K and $[F]$, which reads $\mathsf{K}[F]\alpha \rightarrow [F]\mathsf{K}\alpha$; cf [4], Sect. 1.2, where the Cross Axiom is stated for K and $\Box$. This axiom is so strong that we can dispense with the schemata containing the $\downarrow$-binder then; see [13] for showing this in a simple case.

It is clearly no problem to enlarge the system by taking several epistemic actions into account, i.e., multiple functions $F_1, \ldots, F_n$. But combining these functions in the spirit of deterministic dynamic logic requires additional work. This is postponed to future research.

Unfortunately, the logic PHS is undecidable. This is true since the satisfiability problem of the hybrid system $\mathcal{H}(@, \downarrow)$, see [11], is polynomial time reducible to that of PHS; now, Theorem 37 of [11] applies.

Finally, we relate the issues of this paper to those of the paper [14]. In that paper, the $\downarrow$-binder was considered in connection with subset spaces for the first time. The author introduced, in particular, a binder of the form $\downarrow_x^X$, where $x \in \mathrm{V}_{stat}$ and $X \in \mathrm{V}_{sets}$. That is, both components of the actual neighbourhood situation are always involved in the process of binding. For our purposes, however, it is evidently more suitable to separate bindings. Nevertheless, the operator $\downarrow_x^X$ can be taken for theoretical reasons equally well, whereas a corresponding @-operator is problematic with regard to the neighbourhood situation semantics;[5] see [14], Def. 3. But principally, special emphasis is placed on an appropriate sequent calculus in [14]. Thus the respective approaches not only are distinct regarding the technical details, but also have different goals.

Acknowledgement. I would like to thank Konstantinos Georgatos very much for discussions related to the topic of this paper.

[5] That is why our satisfaction operators are simulated with the aid of the global modality.

References

1. McKinsey, J.C.C.: A solution to the decision problem for the Lewis systems S2 and S4, with an application to topology. Journal of Symbolic Logic 6, 117–141 (1941)
2. McKinsey, J.C.C., Tarski, A.: The algebra of topology. Annals of Mathematics 45, 141–191 (1944)
3. Aiello, M., Pratt-Hartmann, I.E., van Benthem, J.F.A.K.: Handbook of Spatial Logics. Springer, Heidelberg (2007)
4. Dabrowski, A., Moss, L.S., Parikh, R.: Topological reasoning and the logic of knowledge. Annals of Pure and Applied Logic 78, 73–110 (1996)
5. Moss, L.S., Parikh, R., Steinsvold, C.: Handbook of Spatial Logics [3], pp. 299–341. Springer, Heidelberg (2007)
6. Heinemann, B.: Reasoning about knowledge and continuity. In: Goebel, R., Sutcliffe, G. (eds.) Proceedings 19th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2006), pp. 37–42. AAAI Press, Menlo Park (2006)
7. Heinemann, B.: Reasoning about operations on sets. In: Kobti, Z., Wu, D. (eds.) Canadian AI 2007. LNCS, vol. 4509, pp. 308–319. Springer, Heidelberg (2007)
8. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic. Synthese Library, vol. 337. Springer, Heidelberg (2007)
9. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)
10. Heinemann, B.: A hybrid logic for reasoning about knowledge and topology. Journal of Logic, Language and Information 17(1), 19–41 (2008)
11. Areces, C., ten Cate, B.: Handbook of Modal Logic [12], pp. 821–868. Elsevier, Amsterdam (2007)
12. Blackburn, P., van Benthem, J., Wolter, F.: Handbook of Modal Logic. Studies in Logic and Practical Reasoning, vol. 3. Elsevier, Amsterdam (2007)
13. Heinemann, B.: Topological nexttime logic. In: Kracht, M., de Rijke, M., Wansing, H., Zakharyaschev, M. (eds.) Advances in Modal Logic 1, Stanford, CA. CSLI Publications, vol. 87, pp. 99–113. CSLI Publications, Kluwer (1998)
14. Wang, Y.N.: A two-dimensional hybrid logic of subset spaces. In: Ramanujam, R., Sarukkai, S. (eds.) Logic and Its Applications, ICLA 2009. LNCS, vol. 5378, pp. 196–209. Springer, Heidelberg (2009)

Appendix

Proof of Proposition 2. Let s, U be any neighbourhood situation. We must show that

$$(*) \quad s, U \models_{\mathcal{M},g} \downarrow x.\, \downarrow X.\, [F]\mathsf{K} \downarrow y.\, @_{(x \wedge X)} \mathsf{L}[F]y$$

is true. According to Definition 2, $(*)$ is equivalent to

$$s, U \models_{\mathcal{M}, g^{s,U}_{x,X}} [F]\mathsf{K} \downarrow y.\, @_{(x \wedge X)} \mathsf{L}[F]y,$$

where $g^{s,U}_{x,X} := (g^s_x)^U_X$. But this is the case iff

$$F(s), F(U) \models_{\mathcal{M}, g^{s,U}_{x,X}} \mathsf{K} \downarrow y.\, @_{(x \wedge X)} \mathsf{L}[F]y,$$

because of Definition 5. Again by Definition 2, the latter holds iff

$$\text{for all } t \in F(U) : t, F(U) \models_{\mathcal{M}, g^{s,U}_{x,X}} \downarrow y.\, @_{(x \wedge X)} \mathsf{L}[F]y.$$

Substituting the $\downarrow$-operator equivalently yields

$$\text{for all } t \in F(U) : t, F(U) \models_{\mathcal{M}, g^{s,t,U}_{x,y,X}} @_{(x \wedge X)} \mathsf{L}[F]y,$$

where $g^{s,t,U}_{x,y,X}$ is $(g^{s,U}_{x,X})^t_y$. Due to the semantics of the @-operator we obtain

$$\text{for all } t \in F(U) : \exists\, s', U' \in \mathcal{N}_{\mathcal{S}} : s', U' \models_{\mathcal{M}, g^{s,t,U}_{x,y,X}} x \wedge X \wedge \mathsf{L}[F]y$$

as a condition of equal value, which means that

$$\text{for all } t \in F(U) : s, U \models_{\mathcal{M}, g^{s,t,U}_{x,y,X}} \mathsf{L}[F]y.$$

A further application of Definition 2 leads us to

$$\text{for all } t \in F(U) \text{ there is some } u \in U : F(u), F(U) \models_{\mathcal{M}, g^{s,t,U}_{x,y,X}} y$$

so that we finally get

$$\text{for all } t \in F(U) \text{ there is some } u \in U : F(u) \in g^{s,t,U}_{x,y,X}(y) = \{t\},$$

which is obviously true. This completes the proof of $(*)$ and of the proposition thus.

Forcing-Based Cut-Elimination for Gentzen-Style Intuitionistic Sequent Calculus

Hugo Herbelin[1] and Gyesik Lee[2]

[1] INRIA & PPS, Paris Université 7
Paris, France
Hugo.Herbelin@inria.fr
[2] ROSAEC center, Seoul National University
Seoul, Korea
gslee@ropas.snu.ac.kr

Abstract. We give a simple intuitionistic completeness proof of Kripke semantics with constant domain for intuitionistic logic with implication and universal quantification. We use a cut-free intuitionistic sequent calculus as formal system and by combining soundness with completeness, we obtain an executable cut-elimination procedure. The proof, which has been formalised in the Coq proof assistant, easily extends to the case of the absurdity connective using Kripke models with exploding nodes.

Keywords: Intuitionistic Gentzen-style sequent calculus, Kripke semantics, completeness, cut-elimination.

1 Introduction

The intuitionistic completeness proofs for intuitionistic full first-order predicate logic given by Veldman [1] and Friedman [2, Chapter 13] use nonstandard Kripke model and Beth model, respectively (the false formula may be forced at some nodes). Both the proof of Veldman and that of Friedman work by building a model made of infinite contexts. Especially, they had to deal with language extensions and work with spreads in order to meet some closure conditions for disjunction and existential quantification:

- $\Gamma \vdash A \vee B$ implies $\Gamma \vdash A$ or $\Gamma \vdash B$.
- $\Gamma \vdash \exists x\, A(x)$ implies $\Gamma \vdash A(c)$ for some constant c.

Note however that this is not only the case for intuitionistic proofs, but also the case for a classical, Henkin-type proof given in Troelstra and van Dalen [2, Chapter 2].

On the other hand, C. Coquand [3] shows that an intuitionistic proof for intuitionistic propositional logic with implication as a sole logical symbol can be obtained in a much simpler way by building a universal model made of finite contexts of formulae. She gave a mechanised proof of the completeness proof and even got a cut-elimination proof by using some interpreter and inversion functions, a method called "normalisation by evaluation" in general, cf. [4]. They

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 209–217, 2009.

correspond to the soundness and the completeness, respectively, of the propositional logic with implication as sole connective w.r.t. Kripke semantics. The completeness result there is strong in two ways: in the traditional sense that it holds in arbitrary contexts (see [2]) and in a sense (due to Okada [5]) that it builds normal proofs. In this paper, we extend C. Coquand's idea to the intuitionistic first-order predicate logic with implication and universal quantification as logical symbols.

The predicate system used here is a Gentzen-style sequent calculus. The advantage is that the notion of normal proofs is easy to define: one just has to remove the cut rule. More precisely, the calculus we consider is the intuitionistic restriction LJT of a sequent calculus named LKT that Danos *et al* [6] derived from an analytical decomposition of Gentzen's LK within Girard's Linear Logic [7]. LKT is a constrained variant of LK. Its main property is the bijective correspondence between its set of cut-free proofs and the set of normal forms[1] of classical natural deduction [8]. LJT itself is a constrained variant of LJ and its main property is the bijective correspondence between its set of cut-free proofs and the set of normal forms of natural deduction [9,10]. By choosing LJT and its cut-free variant as our reference calculus, we emphasise that our completeness theorem builds not any arbitrary proofs but cut-free ones in a subset of LJ which bijectively maps to normal natural deduction proofs (and hence to normal λ-terms).

We show that the strong completeness holds both for the system with or without $(\bot_i)$. In case of with $(\bot_i)$, we adopt Veldman's *modified* Kripke semantics. Both proofs are intuitionistic and almost the same. Therefore, we can get a very simple cut-elimination proofs as a by-product at the end. A main difference compared with Veldman's or Friedman's proof is that we deal with contexts made of formulae, not just of sentences, therefore we need to handle substitutions.

The Kripke models we are considering are Kripke models with constant domain, i.e. with the domain function D being the same for every world:

$$D = D(w) \text{ for all } w.$$

As a consequence, in the case of universal quantification, considering of all possible future worlds is not necessary and the following simpler definition

$$w \Vdash \forall x\, A(x) \text{ iff, for all } d \in D,\ w \Vdash A(d).$$

gets equivalent, by monotonicity of forcing and invariance of the domain, to the standard definition:

$$w \Vdash \forall x\, A(x) \text{ iff, for all } w' \geq w \text{ and for all } d \in D,\ w' \Vdash A(d).$$

This paper is organised as follows. We first prove the soundness of the Kripke semantics with respect to cut-free LJT. Then give a intuitionistic strong completeness proof which results in the intuitionistic completeness proof. Finally, an intuitionistic cut-elimination process is described.

[1] To be precise: normal forms along a call-by-name reduction semantics.

2 The Sequent Calculus LJT

Let $\mathcal{L} = \mathcal{L}(\mathbb{C}, \mathbb{F}, \mathbb{P})$ be a first-order language with an infinite set $\mathbb{C}$ of individual constants, among them a distinguished constant c_0, a non-empty set $\mathbb{F}$ of functions, and a non-empty set $\mathbb{P}$ of predicates. The logical symbols are the implication $\rightarrow$ and the universal quantification $\forall$. We assume furthermore that there are countably many free variables. Terms, formulae and sentences are defined in the usual way. We follow the convention that $A(x)$ denotes the formula A where the variable x might appear. Any two formulae are considered identical when they are different only in names for bound variables. A variable or a constant is called fresh in a formula A or a context Γ when it does not occur free or at all, respectively.

Definition 1 (Simultaneous substitutions). *Let ρ be a function from finite set of variables to the set of terms.*

1. *Given a term t, $t[.\backslash\rho]$ is inductively defined:*
   - $x[.\backslash\rho] = \begin{cases} \rho(x) & \text{if } x \in \mathit{dom}(\rho), \\ x & \text{otherwise.} \end{cases}$
   - $c[.\backslash\rho] = c$ *for any* $c \in \mathbb{C}$.
   - $(f\, t_1 \cdots t_n)[.\backslash\rho] = f\,(t_1[.\backslash\rho]) \cdots (t_n[.\backslash\rho])$.
2. *Given a formula A, $A[.\backslash\rho]$ is inductively defined:*
   - $(P\, t_1 \cdots t_n)[.\backslash\rho] = P\,(t_1[.\backslash\rho]) \cdots (t_n[.\backslash\rho])$.
   - $(A \rightarrow B)[.\backslash\rho] = (A[.\backslash\rho] \rightarrow (B[.\backslash\rho])$.
   - $(\forall x\, A)[.\backslash\rho] = \forall x\,(A[.\backslash\rho^{-x}])$.

   Here ρ^{-x} denotes the function obtained from ρ with $\mathit{dom}(\rho^{-x}) = \mathit{dom}(\rho)\backslash\{x\}$, i.e., if $y \in \mathit{dom}(\rho)$ and $x \neq y$, then $\rho^{-x}(y) = \rho(y)$ and undefined otherwise. We also take care of variable capture by changing bound variables when necessary.

Given a formula $A(x)$, we use $A_x(t)$ or $A[x\backslash t]$ for $A[.\backslash\rho]$ where $\rho = \{(x, t)\}$. We also consider substitution of a term t for a constant c in a similar way and use the notation $A_c(t)$.

The Gentzen-style sequence calculus LJT is obtained from the intuitionistic sequent calculus LJ by restricting the use of the left introduction rules of the implication and the universal quantification. See Table 1 for the cut-free fragment. In that way, one can get a one-to-one correspondence between cut-free proofs in LJT and normal terms in λ-style calculus.

In LJT, a sequent has one of the forms $\Gamma\,; A \vdash C$ or $\Gamma \vdash C$, where Γ is a list of formulae. That is, the location of a formula occurring multiple times is important. The right side of ";" in the antecedence is called stoup . $\Gamma, \Gamma', \Delta, \ldots$ vary over lists of formulae. We write $A \in \Gamma$ when A occurs in Γ. $\Gamma \sqsubseteq \Delta$ denotes that, for all A, $A \in \Gamma$ implies $A \in \Delta$. $\Gamma_c(t)$ is obtained from Γ by replacing each formula A with $A_c(t)$.

Lemma 2 (Weakening and Exchange). *Let A, C be formulae and Γ, Γ' two contexts such that $\Gamma \sqsubseteq \Gamma'$.*

Table 1. Cut-free LJT

$$\frac{}{\Gamma\,;A\vdash A}\ (Ax) \qquad \frac{\Gamma\,;A\vdash C \quad A\in\Gamma}{\Gamma\vdash C}\ (Contr)$$

$$\frac{\Gamma\vdash A \quad \Gamma\,;B\vdash C}{\Gamma\,;A\to B\vdash C}\ (\to_\ell) \qquad \frac{A,\Gamma\vdash B}{\Gamma\vdash A\to B}\ (\to_r)$$

$$\frac{\Gamma\,;A_x(t)\vdash C}{\Gamma\,;\forall x\,A(x)\vdash C}\ (\forall_\ell) \qquad \frac{\Gamma\vdash A(x) \quad x \text{ fresh in } \Gamma}{\Gamma\vdash \forall x\,A}\ (\forall_r)$$

1. *$\Gamma\vdash C$ implies $\Gamma'\vdash C$.*
2. *$\Gamma\,;A\vdash C$ implies $\Gamma'\,;A\vdash C$.*

Proof. One can easily prove both claims by a simultaneous induction on the deduction. □

Lemma 3. *Let Γ be a context, A, C formulae, and c a constant.*

1. *$\Gamma\vdash C$ implies $\Gamma_c(y)\vdash C_c(y)$ for any variable y which is not bound in Γ, A.*
2. *$\Gamma\,;A\vdash C$ implies $\Gamma_c(y)\,;A_c(y)\vdash C_c(y)$ for any variable y which is not bound in Γ, A, C.*

Proof. By a simple simultaneous induction on deduction. □

The following lemma says that a fresh constant is as good as a fresh variable and will play an important role in the proof of the strong completeness.

Lemma 4. *Given a context Γ, a formula $A(x)$, and a constant c fresh in Γ and $A(x)$, $\Gamma\vdash A_x(c)$ implies $\Gamma\vdash A_x(y)$ for any variable y which is not bound in Γ, A.*

Proof. It follows directly from the lemma just before. □

3 Kripke Semantics

Kripke semantics was created in the late 1950s and early 1960s by Saul Kripke [11,12]. It was first made for modal logic, and later adapted to intuitionistic logic and other non-classical systems. In this section we discuss Kripke models for the first-order predicate logic with implication and universal quantification as sole logical symbols and their connection with intuitionistic validity.

Definition 5. *A Kripke model is a quadruple $\mathcal{K} = (\mathcal{W}, \leq, \Vdash, \mathcal{D}, V)$, $\mathcal{W}$ inhabited, such that*

1. *$(\mathcal{W}, \leq)$ is a partially ordered set.*

2. *$\mathcal{D}$ is an inhabited set, called the* domain *of $\mathcal{K}$.*
3. *Let the language be extended with constant symbols for each element of $\mathcal{D}$. Then $\Vdash$ is a relation between $\mathcal{W}$ and the set of prime sentences in the extended language such that*

$$(w \leq w' \wedge w \Vdash P\, d_1 \cdots d_n) \quad \Rightarrow \quad w' \Vdash P\, d_1 \cdots d_n$$

where $w, w' \in \mathcal{W}$, $P \in \mathbb{P}$, $d_1, ..., d_n \in \mathcal{D}$, and $n = arity(P)$.
4. *V is a function such that*
   - *$V(c) \in \mathcal{D}$ for all $c \in \mathbb{C}$.*
   - *$V(f) : \mathcal{D}^{arity(f)} \to \mathcal{D}$ for all $f \in \mathbb{F}$.*

An association ρ based on $\mathcal{K}$ is a function from a finite set of variables to $\mathcal{D}$. Given an association ρ, each term has an interpretation in $\mathcal{D}$:

- $x\,[\rho] = \begin{cases} \rho(x) & \text{if } x \in \mathit{dom}(\rho), \\ V(c_0) & \text{otherwise.} \end{cases}$
- *$c\,[\rho] = V(c)$ for any $c \in \mathbb{C}$.*
- $(f\, t_1 \cdots t_\ell)\,[\rho] = V(f)(t_1\,[\rho]) \cdots (t_\ell\,[\rho])$.

The forcing relation is then inductively extended by the forthcoming clauses to all $\mathcal{L}$-formulae.

- *$w \Vdash (P\, t_1 \cdots t_n)[\rho]$ iff $w \Vdash P\,(t_1\,[\rho]) \cdots (t_n\,[\rho])$.*
- *$w \Vdash (A \to B)[\rho]$ iff, for all $w' \geq w$, if $w' \Vdash A\,[\rho]$, then $w' \Vdash B\,[\rho]$.*
- *$w \Vdash (\forall x\, A)[\rho]$ iff, for all $d \in D$, $w \Vdash A\,[\rho(x \mapsto d)]$.*

Here $\rho(x \mapsto d)$ denotes the association ρ' such that $\rho'(y) = \rho(y)$ if $y \neq x$ and $\rho'(x) = d$. The definition of forcing is extended to contexts as follows:

$$w \Vdash \Gamma\,[\rho] \text{ iff } w \Vdash A\,[\rho] \text{ for all } A \in \Gamma,$$

Remark 6. There are two points to be mentioned.

1. The forcing relation is upward monotone, i.e., if $w \leq w'$ and $w \Vdash A\,[\rho]$, then $w' \Vdash A\,[\rho]$.
2. As mentioned in the introduction, the forcing definition at the universal quantification case is much simpler than the usual definition, where the domain depends on worlds:

   $w \Vdash (\forall x\, A(x))[\rho]$ iff, for all $w' \geq w$ and $d \in D(w)$, $w' \Vdash A\,[\rho(x \to d)]$.

   Indeed, they are "functionally equivalent" in the sense that soundness and completeness hold in both cases.

We consider a formulation of Kripke semantics for sequent calculus built on two kinds of judgements $\Gamma \vdash C$ and $\Gamma\,; A \vdash C$.

Theorem 7 (Soundness). *We have the following soundness.*

1. *if $\Gamma \vdash C$ then, for all w and ρ, $w \Vdash \Gamma\,[\rho]$ implies $w \Vdash C\,[\rho]$.*
2. *if $\Gamma\,; A \vdash C$ then, for all w and ρ, ($w \Vdash \Gamma\,[\rho]$ and $w \Vdash A\,[\rho]$) implies $w \Vdash C\,[\rho]$.*

Proof. By a simultaneous induction on the deduction. □

4 Completeness

In this section we present a constructive completeness proof by constructing a simple universal model. First we construct a universal model for which a strong completeness holds.

Definition 8 (Universal Kripke model). *The universal Kripke model* $\mathcal{U} = (\mathcal{W}_u, \sqsubseteq, \Vdash_u, \mathcal{D}_u, V_u)$ *is defined as follows:*

- $\mathcal{W}_u$ *is the set of all contexts.*
- $\sqsubseteq$ *denotes the sub-context relation, i.e.,* $\Gamma \sqsubseteq \Gamma'$ *holds when, for all* $A \in \Gamma$, $A \in \Gamma'$.
- $\mathcal{D}_u$ *consists of all closed terms.*
- $V_u(c) = c$ *for all* $c \in \mathbb{C}$, *and* $V_u(f)(t_1, ..., t_\ell) = f\, t_1 \cdots t_\ell$ *for all* $f \in \mathbb{F}$ *and* $t_1, ..., t_\ell \in \mathcal{D}_u^{arity(f)}$
- $\Gamma \Vdash P\, t_1 \cdots t_\ell$ *if* $\Gamma \vdash P\, t_1 \cdots t_\ell$. *It is obvious that* $\Gamma \sqsubseteq \Gamma'$ *and* $\Gamma \vdash P\, t_1 \cdots t_\ell$ *imply* $\Gamma' \vdash t_1 \cdots t_\ell$.

Theorem 9 (Strong Completeness). *Let* Γ *be a context of sentences,* A *a formula, and* ρ *an association based on* $\mathcal{K}_u$ *such that* $FV(A) \subseteq \mathsf{dom}(\rho)$. *Then*

1. *If* $\Gamma \Vdash A\,[\rho]$ *then* $\Gamma \vdash A[.\backslash\rho]$.
2. *If, for all formula* C *and context* Γ' *such that* $\Gamma \sqsubseteq \Gamma'$, $\Gamma'; A[.\backslash\rho] \vdash C$ *implies* $\Gamma' \vdash C$, *then it holds that* $\Gamma \Vdash A\,[\rho]$.

Proof. Given the assumptions we prove both claims by a simultaneous induction on the complexity of A. Note first that $t\,[\rho] = t[.\backslash\rho]$ for any term t occurring in A since $FV(A) \subseteq \mathsf{dom}(\rho)$.

1. case: A is a prime formula. Then the first claim is obvious. For the second claim take just $\Gamma' := \Gamma$ and $C := A[.\backslash\rho]$.

2. case: $A = A_1 \to A_2$.

- Assume $\Gamma \Vdash (A_1 \to A_2)\,[\rho]$. To show $\Gamma \vdash A_1[.\backslash\rho] \to A_2[.\backslash\rho]$, it suffices to prove that $A_1[.\backslash\rho], \Gamma \vdash A_2[.\backslash\rho]$. Note that the i.h. on A_1 for the second claim implies that $A_1[.\backslash\rho], \Gamma \Vdash A_2\,[\rho]$. Indeed, the premise of the second claim holds trivially for any Γ' such that $A_1[.\backslash\rho], \Gamma \sqsubseteq \Gamma'$. This in turn implies that $A_1[.\backslash\rho], \Gamma \Vdash A_2\,[\rho]$ by the assumption. Then the i.h. on A_2 for the first claim leads to the goal.
- Assume for all formula C and context Γ' such that $\Gamma \sqsubseteq \Gamma'$, $\Gamma'; A_1[.\backslash\rho] \to A_2[.\backslash\rho] \vdash C$ implies $\Gamma' \vdash C$. Assume furthermore that $\Gamma \sqsubseteq \Delta$ and $\Delta \Vdash A_1\,[\rho]$. Then it remains to show $\Delta \Vdash A_2\,[\rho]$. For that we apply the i.h. on A_2 for the second claim. Let C be a formula and Δ' a context such that $\Delta \sqsubseteq \Delta'$, assume $\Delta'; A_2[.\backslash\rho] \vdash C$. Note that $\Delta \vdash A_1[.\backslash\rho]$ by i.h. on A_1 for the first claim, hence $\Delta' \vdash A_1[.\backslash\rho]$ by the Weakening Lemma 2. By applying $(\to_\ell)$ we get $\Delta'; (A_1 \to A_2)[.\backslash\rho] \vdash C$, so $\Delta' \vdash C$ holds by the assumption.

3. case: $A = \forall x\, B(x)$.

- Assume $\Gamma \Vdash (\forall x\, B)\,[\rho]$, i.e., for all closed term t, $\Gamma \Vdash B\,[\rho(x \mapsto t)]$. To show $\Gamma \vdash \forall x\,(B[.\backslash\rho^{-x}])$, we need to prove that $\Gamma \vdash (B[.\backslash\rho^{-x}])[x\backslash y]$ for some variable y fresh in Γ and $B[.\backslash\rho^{-x}]$. For this we show $\Gamma \vdash (B[.\backslash\rho^{-x}])[x\backslash c]$ for some constant c and apply Lemma 4. Let c be a constant fresh in Γ and $B[.\backslash\rho^{-x}]$. Note first that

$$(B[.\backslash\rho^{-x}])[x\backslash c] = B[.\backslash\rho^{-x}(x \mapsto c)] = B[.\backslash\rho(x \mapsto c)]$$

  since the values of ρ are closed terms. However, $\Gamma \vdash B[.\backslash\rho(x \mapsto c)]$ follows from the i.h. on B for the first claim.[2]
- Assume for all formula C and context Γ' such that $\Gamma \sqsubseteq \Gamma'$, it holds that $\Gamma'\,; \forall x\,(B[.\backslash\rho^{-x}]) \vdash C$ implies $\Gamma' \vdash C$. Given a closed term t, we have to show that $\Gamma \Vdash B\,[\rho(x \mapsto t)]$. In order to apply the i.h. on B for the second claim assume furthermore that a formula C and a context Γ' are given such that $\Gamma \sqsubseteq \Gamma'$ and $\Gamma'\,; B[.\backslash\rho(x \mapsto t)] \vdash C$. Note that $B[.\backslash\rho(x \mapsto t)] = B[.\backslash\rho^{-x}][x\backslash t]$ since only closed terms are substituted. Therefore, it holds that $\Gamma'\,; \forall x\,(B[.\backslash\rho^{-x}]) \vdash C$. Then by the main assumption, $\Gamma' \vdash C$. The i.h. implies $\Gamma \Vdash B\,[\rho(x \mapsto t)]$. □

Corollary 10. *For any context Γ of sentences, $\Gamma \Vdash \Gamma$.*

Proof. The second claim of the strong completeness and the rule (*Contr*) implies that $\Gamma \Vdash A$ for any $A \in \Gamma$. □

Theorem 11 (Completeness). *Let Γ be a context of sentences and A a sentence. If for all Kripke model $\mathcal{K}$ and a world w in $\mathcal{K}$, $w \Vdash \Gamma$ implies $w \Vdash A$, then $\Gamma \vdash A$.*

Proof. It follows from the Strong Completeness and the fact that $\Gamma \Vdash \Gamma$, i.e., $\Gamma \vdash A$ iff $\Gamma \Vdash A$. □

Corollary 12. *In the intuitionistic predicate logic with implication and universal quantification as sole connectives, the Kripke semantics with constant domain is* functionally equivalent *to the usual Kripke semantics where the domain depends on worlds in the sense that soundness and completeness hold in both cases.*

Proof. The exactly same proofs for soundness and completeness hold with the usual Kripke models. □

Remark 13. We furthermore believe that, in the first-order predicate language with $\rightarrow$, $\wedge$ and $\forall$ as sole connectives, the usual Kripke semantics is equivalent to the Kripke semantics with constant domain in the sense that each usual Kripke model can be transformed into an equivalent Kripke model with a constant domain.

[2] One can see here that we don't need any quantification in the definition of $w \Vdash \forall x\, A(x)$.

Remark 14. The completeness above easily extends to the case of the absurdity connective

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash C}\ (\bot_i)$$

using modified Kripke models with exploding nodes à la Veldman.

A modified Kripke model $\mathcal{K} = (\mathcal{W}, \leq, \Vdash, \mathcal{D}, V)$ is defined as the (unmodified) Kripke model, but with one change:

– $w \Vdash (P\, t_1 \cdots t_n)[\rho]$ iff $(w \Vdash P\,(t_1\,[\rho]) \cdots (t_n\,[\rho])$ or $w \Vdash \bot)$.

The universal Kripke model is defined in the same way as before, but with the following additional clause:

– $\Gamma \Vdash \bot$ iff $\Gamma \vdash \bot$.

Then nothing new is involved in the proof of completeness, the construction proceeds as before. Note only that $\Gamma \vdash \bot$ implies $\Gamma \vdash A$ for any formula A.

On the other hand, if we include the absurdity rule and want to stick to the (unmodified) Kripke semantics, we have to give up constructiveness in the proof above. This is because we have to deal only with consistent context Γ, i.e., $\Gamma \nvdash \bot$. However, it is in general undecidable to check if a context is consistent or not. In the implication case of the strong completeness, we had to make case distinction between $A_1[.\backslash\rho], \Gamma \vdash \bot$ or not. This maybe is not so surprising because, in the full first-order predicate logic, an intuitionistic completeness proof entails Markov's Principle, a non-intuitionistic principle, see Kreisel [13].

5 Cut Admissibility

In this section we consider only sentences and contexts of sentences. Then the cut-rule is admissible with/without $(\bot_i)$.

Theorem 15 (Cut admissibility). *$\Gamma \vdash A$ and $\Gamma; A \vdash B$ imply $\Gamma \vdash B$.*

Proof. By the Strong Completeness it suffices to show $\Gamma \Vdash B$. But this follows from the Soundness applied on the two assumptions. Note also that $\vdash$ denotes a cut-free system. □

6 Conclusion

We extended C. Coquand's proof of soundness and completeness for implicative natural deduction w.r.t. Kripke semantics [3] to the case of predicate logic with implication and universal quantification. We could show that omitting disjunction and existential quantification from the intuitionistic first-order predicate logic results, as it was the case for C. Coquand, in a significantly simple, intuitionistic completeness proof with respect to (also simplified) Kripke semantics.

The fact that all of the proofs given in this paper are intuitionistic has been verified in the proof assistant Coq, cf. [14]. Indeed, the whole work is formalised,

so that we can get a mechanical process producing cut-free proofs. The formalisation is performed using cofinite quantification for fresh variables and a locally named approach with two kinds of names for variables, one for free variables and the other for binders. The formalisation is publicly available from the web page of the first author, directory code/kripke.

References

1. Veldman, W.: An intuitionistic completeness theorem for intuitionistic predicate logic. J. Symb. Log. 41(1), 159–166 (1976)
2. Troelstra, A.S., van Dalen, D.: Constructivism in Mathematics: An Introduction I and II. Studies in Logic and the Foundations of Mathematics, vol. 121, 123. North-Holland, Amsterdam (1988)
3. Coquand, C.: From Semantics to Rules: A Machine Assisted Analysis. In: Meinke, K., Börger, E., Gurevich, Y. (eds.) CSL 1993. LNCS, vol. 832, pp. 91–105. Springer, Heidelberg (1994)
4. Berger, U., Eberl, M., Schwichtenberg, H.: Normalisation by Evaluation. In: Möller, B., Tucker, J.V. (eds.) NADA 1997. LNCS, vol. 1546, pp. 117–137. Springer, Heidelberg (1998)
5. Okada, M.: A uniform semantic proof for cut-elimination and completeness of various first and higher order logics. Theor. Comput. Sci. 281(1-2), 471–498 (2002)
6. Danos, V., Joinet, J.B., Schellinx, H.: LKQ and LKT: sequent calculi for second order logic based upon dual linear decompositions of the classical implication. In: Advances in Linear Logic, vol. 222, pp. 211–224. Cambridge University Press, Cambridge (1995)
7. Girard, J.Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
8. Parigot, M.: Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In: Voronkov, A. (ed.) LPAR 1992. LNCS, vol. 624, pp. 190–201. Springer, Heidelberg (1992)
9. 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)
10. Herbelin, H.: Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes. Ph.D. thesis, Université Paris 7 (January 1995)
11. Kripke, S.: A Completeness Theorem in Modal Logic. J. Symb. Log. 24(1), 1–14 (1959)
12. Kripke, S.: Semantical considerations on modal and intuitionistic logic. Acta Philos. Fennica 16, 83–94 (1963)
13. Kreisel, G.: On Weak Completeness of Intuitionistic Predicate Logic. J. Symb. Log. 27(2), 139–158 (1962)
14. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development (Coq'Art: The Calculus of Inductive Constructions). EATCS Texts in Theoretical Computer Science, vol. XXV. Springer, Heidelberg (2004)

Property Driven Three-Valued Model Checking on Hybrid Automata

Kerstin Bauer, Raffaella Gentilini, and Klaus Schneider

University of Kaiserslautern, Department of Computer Science, Germany
{k_bauer,gentilin,schneider}@cs.uni-kl.de

Abstract. In this paper, we present a three-valued *property driven* model checking algorithm for the logic CTL on hybrid automata. The technique of multi-valued model checking for hybrid automata aims at combining the advantages of classical methods based either on the preorder of simulation or on bounded reachability. However, as originally defined, it relies on the preliminary definition of special abstractions for combined over- and under-approximated reachability analysis, whose size is crucial and can be infinite. Our procedure avoids the above problem, since it is based on an incremental construction of the abstraction for the original hybrid automaton, that is suitably driven by the property under consideration.

1 Introduction

Hybrid automata [11,1] provide an appropriate modeling paradigm for systems where continuous variables interact with discrete modes. Such models are frequently used in complex engineering fields like embedded systems, robotics, automotive industries, avionics, and aeronautics [2,18,9]. In hybrid automata, the interaction between discrete and continuous dynamics is naturally expressed by associating a set of differential equations to every location of a finite automaton.

Finite automata and differential equations are well established formalisms in mathematics and computer science. Despite of their long-standing tradition, their combination in form of hybrid automata leads to surprisingly difficult problems that are often undecidable. In particular, the *reachability* problem is undecidable for most families of hybrid automata [14,15,16,10,1,5], and the few decidability results are built upon strong restrictions of the dynamics [3,12]. The reachability analysis of hybrid automata is a fundamental task, since checking *safety* properties of the underlying system can be reduced to a reachability problem for the set of bad configurations [11].

For this reason, a growing body of research is being developed on the issue of dealing with approximated reachability on undecidable – yet reasonably expressive – hybrid automata [6,19,8,17,18]. To this end, most of the techniques proposed so far either rely on bounded state-reachability or on the definition of finite abstractions. While the first approach suffers inherently of incompleteness, the quest for *soundness* is a key issue in the context of methods based on abstractions. In fact, abstractions can introduce unrealistic behaviors that may yield to spurious errors being reported in the safety analysis. Usually, a simulation preorder is required to relate the abstraction to the concrete

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 218–229, 2009.

dynamics of the hybrid system under consideration, ensuring at least the correctness of each response of *non*-reachability. In general, the simulation preorder from the abstraction to the hybrid automaton allows for preservation of only *true* formulas in the *universal fragment* of a branching temporal logic. Recently, a novel framework has been proposed [7,4], featuring the capability of *both proving and disproving* properties expressed by means of the logic CTL or even the mu-calculus on (undecidable) hybrid automata. Such a method is based on the definition of a sound three-valued semantics [13] for the considered logics over so-called discrete bounded bisimulation (DBB) abstractions. Unfortunately, the size of the DBB quotients is not guaranteed to be finite for general hybrid systems. In fact, the method in [7,4] was applied to the only (undecidable) family of fully o-minimal hybrid automata.

In this paper, we sharpen the results concerning the size of the DBB abstractions on different classes of hybrid automata. Moreover, we design a new framework for three-valued model checking of the logic CTL on (undecidable) hybrid automata. Such a method has the key advantage of avoiding the a-priori construction of DBB abstractions that could result into an infinite partitioning. Rather, the abstraction is built on-the-fly and is driven by the ongoing three-valued model checking task.

2 Preliminaries

In this section, we introduce the basic definitions and the notations used in the paper.

Definition 1 (Hybrid Automata [3]). *A* hybrid automaton *is a tuple* $H = (L, E, X,$ *Init*, *Inv*, $F, G, R)$ *with the following components:*

- *a finite set of* locations L
- *a finite set of* discrete transitions $E \subseteq L \times L$
- *a finite set of* continuous variables $X = \{x_1, \ldots x_n\}$ *that take values in* $\mathbb{R}$
- *an initial set of conditions: Init* $\subseteq L \times \mathbb{R}^n$
- *an* invariant location labeling *Inv:* $L \mapsto 2^{\mathbb{R}^n}$
- *a function* $F : L \times \mathbb{R}^n \mapsto 2^{\mathbb{R}^n}$ *assigning to each location* $\ell \in L$ *a differential rule defining the evolution of continuous variables within* ℓ
- *a* guard edge labeling $G : E \mapsto 2^{\mathbb{R}^n}$
- *a* reset edge labeling $R : E \times \mathbb{R}^n \mapsto 2^{\mathbb{R}^n}$

We write $\mathbf{v}$ to represent a valuation $(v_1, \ldots, v_n) \in \mathbb{R}^n$ of the variables' vector $\mathbf{x} = (x_1, \ldots, x_n)$, whereas $\dot{\mathbf{x}}$ denotes the first derivatives of the variables in $\mathbf{x}$ (they all depend on the time, and are therefore rather functions than variables). A *state* in H is a pair $s = (\ell, \mathbf{v})$, where $\ell \in L$ is called the *discrete component* of s and $\mathbf{v}$ is called the *continuous component* of s. A *run* of $H = (L, E, X,$ *Init*, *Inv*, $F, G, R)$, starts at any $(\ell, \mathbf{v}) \in$ *Init* and consists of continuous evolutions (within a location) and discrete transitions (between two locations). Formally, a run of H is a path with alternating continuous and discrete steps in the *time abstract transition system* of H, defined below:

Definition 2. *The* time abstract transition system *of the hybrid automaton* $H = (L, E,$ $X,$ *Init*, *Inv*, $F, G, R)$ *is the transition system* $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$, *where:*

- $Q \subseteq L \times \mathbb{R}^n$ *and* $(\ell, \mathbf{v}) \in Q$ *if and only if* $\mathbf{v} \in \mathit{Inv}(\ell)$
- $Q_0 \subseteq Q$ *and* $(\ell, \mathbf{v}) \in Q_0$ *if and only if* $\mathbf{v} \in \mathit{Init}(\ell) \cap \mathit{Inv}(\ell)$
- $\ell_{\rightarrow} = E \cup \{\delta\}$ *is the set of edge labels*
- $\rightarrow \subseteq Q \times \ell_{\rightarrow} \times Q$ *is the set of labeled transitions, that are determined as follows:*
  - *there is a* continuous transition $(\ell, \mathbf{v}) \xrightarrow{\delta} (\ell, \mathbf{v}')$*, if and only if there is a differentiable function* $f : [0,t] \rightarrow \mathbb{R}^n$*, with* $\dot{f} : [0,t] \rightarrow \mathbb{R}^n$ *such that:*
    1. $f(0) = \mathbf{v}$ *and* $f(t) = \mathbf{v}'$
    2. *for all* $\varepsilon \in (0,t)$*,* $f(\varepsilon) \in \mathit{Inv}(\ell)$*, and* $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$.
  - *there is a* discrete transition $(\ell, \mathbf{v}) \xrightarrow{e} (\ell', \mathbf{v}')$ *if and only if there exists an edge* $e = (\ell, \ell') \in E$*,* $\mathbf{v} \in G(\ell)$ *and* $\mathbf{v}' \in R((\ell, \ell'), \mathbf{v})$.

A region is a subset of the states Q of $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$. Given a region B and a transition label $a \in \ell_{\rightarrow}$, the predecessor region $pre_a(B)$ is defined as the region $\{q \in Q \mid \exists q' \in B.\ q \xrightarrow{a} q'\}$. The *bisimulation* and the *simulation* relations are two fundamental tools in the context of hybrid automata abstraction.

Definition 3 ((Bi)simulation). *Let* $T_1 = \langle Q^1, Q_0^1, \ell_{\rightarrow}, \rightarrow^1 \rangle$*,* $T_2 = \langle Q^2, Q_0^2, \ell_{\rightarrow}, \rightarrow^2 \rangle$*,* $Q^1 \cap Q^2 = \emptyset$*, be two labeled transition systems*[1] *and let* $\mathcal{P}$ *be a partition on* $Q_1 \cup Q_2$*. A* simulation *from* T_1 *to* T_2 *is a non-empty relation on* $\rho \subseteq Q^1 \times Q^2$ *such that, for all* $(p, q) \in \rho$*:*

- $p \in Q_0^1$ *iff* $q \in Q_0^2$ *and* $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$*, where* $[p]_{\mathcal{P}}$ *(*$[q]_{\mathcal{P}}$*) is the class of* p *(*q*) w.r.t.* $\mathcal{P}$*.*
- *for each label* $a \in \ell_{\rightarrow}$*, if there exists* p' *such that* $p \xrightarrow{a} p'$*, then there exists* q' *such that* $(p', q') \in \rho$ *and* $q \xrightarrow{a} q'$*.*

If there exists a simulation from T_1 *to* T_2*, then we say that* T_2 simulates T_1*, denoted* $T_1 \prec_S T_2$*. If* $T_1 \prec_S T_2$ *and* $T_2 \prec_S T_1$*, then* T_1 *and* T_2 *are said* similar*, denoted* $T_1 \equiv_S T_2$*. If* ρ *is a simulation from* T_1 *to* T_2*, and* ρ^{-1} *is a simulation from* T_2 *to* T_1*, then* ρ *is said a* bisimulation*.*

Definition 4 formally introduces the concept of abstraction for hybrid automata, on the ground of the notion of simulation.

Definition 4. *Let* $H = (L, E, X, \mathit{Init}, \mathit{Inv}, F, G, R)$ *be a hybrid automaton. An abstraction of* H *is a finite labeled transition system* $\mathcal{A} = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \rightarrow \rangle$ *where:*

- $Q_{\mathcal{A}}$ *is a finite partition of* Q*,* $Q_{\mathcal{A}}^0$ *is a finite partition of* Q_0*,* $\ell_{\rightarrow} = \{\delta\} \cup E$*, and* $\rightarrow \subseteq Q_{\mathcal{A}} \times Q_{\mathcal{A}}$*.*
- $\mathcal{A}^* = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta *} \rangle$ *simulates*[2] T_H

Definition 6 recapitulates the semantics of the temporal logic CTL (where the neXt temporal operator is intensionally omitted because of the density of the underlying time framework) on hybrid automata [1,11].

Definition 5 (CTL for Hybrid Automata). *Let* AP *be a finite set of propositional letters and* $\mathsf{p} \in \mathsf{AP}$*.* CTL *is the set of formulas defined by the following grammar rules:*

$$\phi ::= \mathsf{p} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{E}(\phi_1 \mathsf{U} \phi_2) \mid \mathsf{A}(\phi_1 \mathsf{U} \phi_2)$$

[1] A labeled transition system is a quadruple $\langle Q, Q_0, \ell_{\rightarrow}, \rightarrow \rangle$, where $Q_0 \subseteq Q$, $\rightarrow \subseteq Q \times \ell_{\rightarrow} \times Q$.
[2] Given a relation $\rightarrow$, we denote by $\rightarrow^*$ its transitive and reflexive closure.

Definition 6 (CTL Semantics). *Let $H = (L, E, X, Init, Inv, F, G, R)$ be a hybrid automaton, let Q (Q_0) be he set of states (initial states) of H, and let AP be a set of propositional letters. Consider $\ell_{\mathsf{AP}} : L \times \mathbb{R}^n \mapsto 2^{\mathsf{AP}}$. Given $\phi \in \mathsf{CTL}$ and $s \in Q$, $s \models \phi$ is inductively defined as follows:*

- $s \models \mathsf{p}$ *if and only if* $\mathsf{p} \in \ell_{\mathsf{AP}}(s)$
- $s \models \neg\phi$ *if and only if not* $s \models \phi$
- $s \models \phi_1 \wedge \phi_2$ *if and only if* $s \models \phi_1$ *and* $s \models \phi_2$
- $s \models \mathsf{E}(\phi_1 \mathsf{U} \phi_2)$ *if and only if there exists a run ρ and a time t such that:*
  - $\rho(t) \models \phi_2$
  - $\forall t' \leq t.\ \rho(t') \models \phi_1$
- $s \models \mathsf{A}(\phi_1 \mathsf{U} \phi_2)$ *if and only if for each run ρ there exists a time t such that:*
  - $\rho(t) \models \phi_2$
  - $\forall t' \leq t.\ \rho(t') \models \phi_1$

Then, $H \models \phi$ iff $\forall s \in Q_0.\ s \models \phi$.

3 Three-Valued Model Checking on Hybrid Automata and the Problem of Discrete Bounded Bisimulation Explosion

In this section we recall the notion of *discrete bounded bisimulation* abstraction on hybrid automata [7] and we sharpen the results relative to the size of its quotient on different families of hybrid systems. Discrete bounded bisimulation (cf. Definition 7) was introduced in [7] as a bisimulation approximation for hybrid automata, suitable to support sound *three*-valued model checking results on these systems.

Definition 7 (Discrete Bounded Bisimulation (DBB)). *Consider the time abstract transition system $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$ of a hybrid automaton H, and let $\mathcal{P}$ be a partition on Q:*

1. *$\equiv_0 \in Q \times Q$ is the maximum relation on Q such that for all $p, q \in Q$:*
   if $p \equiv_0 q$ then (a) $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$ *and* $p \in Q_0$ *iff* $q \in Q_0$
   (b) $\forall p'.\ p \xrightarrow{\delta} p' \Rightarrow \exists q'.\ p' \equiv_0 q' \wedge q \xrightarrow{\delta} q'$
   (c) $\forall q'.\ q \xrightarrow{\delta} q' \Rightarrow \exists p'.\ p' \equiv_0 q' \wedge p \xrightarrow{\delta} p'$
2. *Given $n \in \mathbb{N}^+$, $\equiv_n$ is the maximum relation on Q such that for all $p, q \in Q$:*
   if $p \equiv_n q$ then (a) $p \equiv_{n-1} q$
   (b) $\forall p'.\ p \xrightarrow{\delta} p' \Rightarrow \exists q'.\ p' \equiv_n q' \wedge q \xrightarrow{\delta} q'$
   (c) $\forall q'.\ q \xrightarrow{\delta} q' \Rightarrow \exists p'.\ p' \equiv_n q' \wedge p \xrightarrow{\delta} p'$
   (d) $\forall p'.\ p \xrightarrow{e} p' \Rightarrow \exists q'.\ p' \equiv_{n-1} q' \wedge q \xrightarrow{e} q'$
   (e) $\forall q'.\ q \xrightarrow{e} q' \Rightarrow \exists p'.\ p' \equiv_{n-1} q' \wedge p \xrightarrow{e} p'$

Given $n \in \mathbb{N}$, the relation $\equiv_n$ will be referred to as n-DBB equivalence.

In [7] the notion of DBB was shown to provide finite abstractions on the family of fully o-minimal hybrid automata, for which the reachability problem is not decidable [14]. On this ground, it was possible to design a three-valued model checking algorithm for

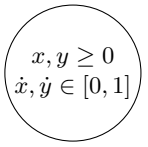


Fig. 1. Single location rectangular automaton

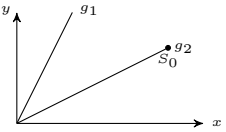


Fig. 2. Initial partition of the state space

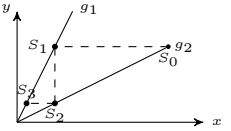


Fig. 3. Infinite bisimulation computation

the logic CTL and the μ-calculus on fully o-minimal hybrid systems. Unfortunately, there are other important families of undecidable hybrid automata for which such a finiteness result does not hold. In particular, the following example (cf. Example 1) considers the class of (non-initialized) rectangular automata (RHA) for which the differential inclusions describing the continuous dynamics are *rectangular*, i.e. given by

$$\bigwedge_{i=1\ldots n} \dot{x}_i \in [a_i, b_i], a_i, b_i \in \mathbb{R}^+$$

RHA are well known to be undecidable w.r.t. the reachability problem [12], unless strong constraints such as the so-called *initialization*[3] are considered. Example 1 proves that the size of DBB abstractions cannot be guaranteed to be finite on RHA. Thus, the (DBB based) framework for the three-valued model checking of hybrid automata developed in [7] can not be applied to RHA.

Example 1 (DBB explosion for RHA). Consider the simple rectangular automaton given in Figure 1. Figure 2 provides an initial partition over the states-space of H, given by the line $g_1 : y = 2x$, the point $S_0 = (8, 4)$, and the segment $g_2 : y = \frac{1}{2}x \wedge 0 \leq x < 8$. Figure 3 depicts the construction procedure of the 0-DBB abstraction over H, w.r.t. the initial partition given in Figure 2. The procedure does not terminate since the regions g_1, g_2 need to be split into the infinite set of regions $\{(S_{2i+3}, S_{2i+1}]\}_{i\geq 0}, \{(S_{2i+2}, S_{2i}]\}_{i\geq 0}$, with $S_{2i+1} = (x_{2i}, \frac{1}{4}y_{2_i})$ and $S_{2i} = (\frac{1}{4}x_{2i-1}, y_{2i-1})$due to the point $S_0 = (8, 4)$.

4 A General Algorithmic Framework for Property Driven Three-Valued CTL Model Checking on Hybrid Automata

In this Section, we develop a new algorithmic approach to the three-valued model checking of hybrid automata. Rather than relying on a preliminary static abstraction of the considered dynamical system, our method features a dynamic partitioning process, which is suitably driven by the three-valued model checking of the formula given as input. In other words, the three-valued verification task is accomplished *on the fly*, thus avoiding the problems related to the (possibly infinite) size of the DBB-abstraction.

As shown in Figure 4, our algorithm PROPERTYDRIVEN3MC accomplishes its task upon the subsequent usage of (1) the discrete switches (within the procedure D3MC, called at line 7) and (2) the continuous dynamics of the hybrid automaton given as input

[3] RHA are initialized, iff for each continuous variable during any discrete transition either the continuous flow does not change or the value of the variable is reinitialized.

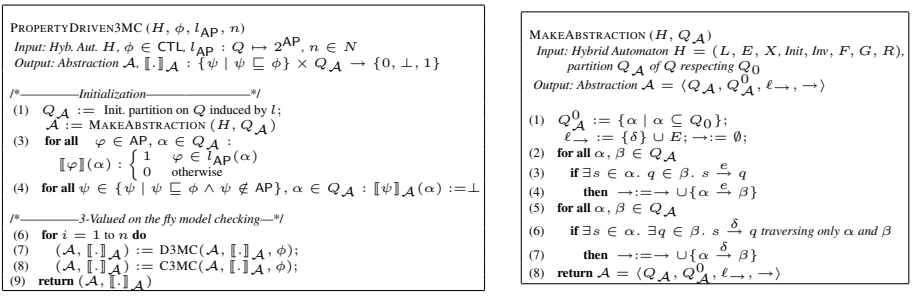


Fig. 4. The Property Driven Three-Valued Model Checking Procedure

(within the procedure C3MC, called at line 8). Both discrete and continuous dynamics are employed to grow dynamically an abstraction $\mathcal{A}$ having the following key property. On the one hand, $\mathcal{A}$ can be *globally* viewed as an *over*-approximation of the dynamics of the original hybrid automaton H. On the other hand, one can enucleate *local* under-approximations of the evolutions in H. These features permit to perform our on the fly three-valued model checking task. In particular, underapproximations (overapproximations) can be suitably used to deal with existential (universal) subformulas.

4.1 The Procedure D3MC

Within each iteration of the main algorithm, the procedure D3MC in Figure 5 exploits the discrete dynamics of the underlying hybrid automaton to (1) refine a growing abstraction $\mathcal{A}$, and (2) proceed with its on the fly three-valued model checking. First, each discrete edge is used to split those classes in the current partition, for which some (sub)-formula of interest evaluates to the third value $\bot$ (lines (2)–(4)). Such a split allows to separate set of states that *must* evolve to different regions via a discrete switch, and therefore should be assigned distinguished model checking values. Then, the paths through $\mathcal{A}$ are used to suitably *propagate* the current information, in order to infer new sound true or false evaluations of the (sub)-formulas of interest. This is done within the subprocedure D3MCFORM, called at line 7. More in detail, D3MCFORM

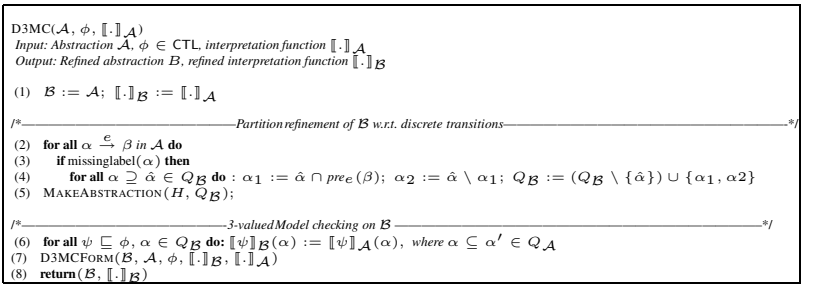


Fig. 5. The procedure D3MC within the algorithm PROPERTYDRIVEN3MC

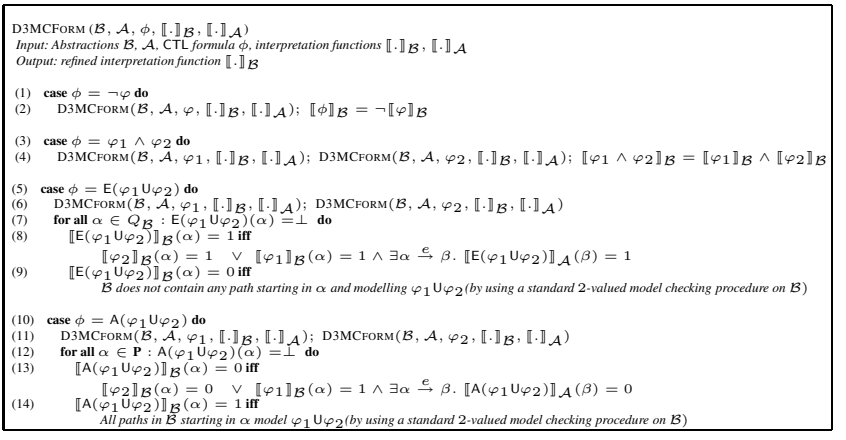


Fig. 6. The subprocedure D3MCFORM, used within the algorithm PROPERTYDRIVEN3MC

(cf. Figure 6) refines the interpretation function following different schemes, depending on the existential (resp. universal) character of the considered operator. For existential formulas, only paths consisting of a single discrete switch are followed, since the latter provide (local) under-approximations of the concrete evolutions (lines 5–9). For universal formulas instead, global paths are followed (lines 10–14).

On the ground of Lemma 1, Lemma 2 below states formally the correctness of each on the fly model checking macro-step based on the usage of discrete switches, and performed within the procedure D3MC. Consider a call to D3MC$(\mathcal{A}, \phi, [\![\cdot]\!]_{\mathcal{A}})$, where ϕ is a CTL formula, $\mathcal{A}$ is an abstraction of the hybrid automaton H, and $[\![\cdot]\!]_{\mathcal{A}} : \Gamma = \{\varphi \mid \varphi \sqsubseteq \phi\} \times Q_{\mathcal{A}} \mapsto \{0, 1, \bot\}$[4]. Moreover, let from now on $\preceq$ be the *information ordering* on the set of truth values $\{\bot, 0, 1\}$, in which $\bot \preceq 1$, $\bot \preceq 0$, and $x \preceq x$ (for all $x \in \{\bot, 0, 1\}$).

Lemma 1. *The abstraction refinement $\mathcal{B}$ of $\mathcal{A}$ produced upon the execution of the procedure* D3MC$(\mathcal{A}, \phi, [\![\cdot]\!])$ *is such that*[5]*:*

1. $T_H \prec_S \mathcal{B}^* \prec_S \mathcal{A}^*$.
2. *If $\alpha \xrightarrow{e} \beta$ in $\mathcal{B}$, then each state in α has a discrete successor in the unique class of $\mathcal{A}$ containing β.*

Lemma 2. *Consider* D3MC$(\mathcal{A}, \phi, [\![\cdot]\!]_{\mathcal{A}})$ *and assume that $[\![\cdot]\!]_{\mathcal{A}}$ is sound w.r.t. the model checking of ϕ on H, i.e: $\forall \alpha \in Q_{\mathcal{A}}.\ \forall \varphi \sqsubseteq \phi.\ [\![\varphi]\!]_{\mathcal{A}}(\alpha = [s]) \preceq [\![\varphi]\!]_H(s)$*
Then, the refined interpretation function $[\![\cdot]\!]_{\mathcal{B}}$ computed by D3MC$(\mathcal{A}, \phi, [\![\cdot]\!]_{\mathcal{A}})$ *satisfies:*

$$\forall \alpha \in Q_{\mathcal{B}}.\ \forall \varphi \sqsubseteq \phi.\ [\![\phi]\!]_{\mathcal{A}}(\hat{\alpha}) \preceq ([\![\varphi]\!]_{\mathcal{B}}(\alpha = [s]) \preceq [\![\varphi]\!]_H(s), \text{ where } \alpha \subseteq \hat{\alpha} \in \mathcal{A}$$

[4] We use Kleene's definition of three-valued logic [13].

[5] Recall that given an abstraction $\mathcal{A} = \langle Q_{\mathcal{A}}, Q^0_{\mathcal{A}}, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta} \rangle$ for the hybrid automaton H, we denote by $\mathcal{A}^*$ the structure $\mathcal{A}^* = \langle Q_{\mathcal{A}}, Q^0_{\mathcal{A}}, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta}{}^* \rangle$, according to Definition 4.

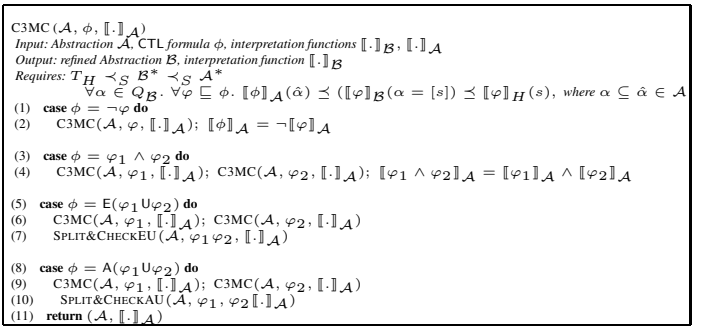


Fig. 7. The subprocedure C3MC within the algorithm PROPERTYDRIVEN3MC

4.2 The Subprocedure C3MC

The purpose of the procedure C3MC in our main algorithm is that of gaining information from the continuous dynamics of the given hybrid automaton to proceed – on the fly – in the three-valued model checking of the considered property. The continuous dynamics associated to hybrid automata can be governed by differential inclusions (like in RHA) or by differential equations, the latter defined in a variety of theories over the reals (linear [11], semi-algebraic [6,17], o-minimal [14]). To cope with such a variety of patterns, we proceed bottom-up by defining first a simple interface for our procedure C3MC (cf. Figure 7). Within such an interface boolean formulae are dealt with using standard techniques. Instead, the resolution of temporal operators is only required to be implemented by adhering to the following constraints: If $\langle \mathcal{B}, [\![\cdot]\!]_{\mathcal{B}} \rangle$ are the refined abstraction and the interpretation function computed by the function SPLIT&CHECKEU (resp. SPLIT&CHECKAU) then

- $T_H \prec_S \mathcal{B}^* \prec_S \mathcal{A}^*$
- $\forall \alpha \in Q_{\mathcal{B}}.\ \forall \varphi \sqsubseteq \phi.\ [\![\phi]\!]_{\mathcal{A}}(\hat{\alpha}) \preceq ([\![\varphi]\!]_{\mathcal{B}}(\alpha = [s]) \preceq [\![\varphi]\!]_H(s),$ *where* $\alpha \subseteq \hat{\alpha} \in \mathcal{A}$

In the next section, we will propose various implementations of the interface for the procedure C3MC, specialized for different classes of hybrid automata. Theorem 1, states the correctness of the main algorithm PROPERTYDRIVEN3MC, assuming a sound implementation of the interface for C3MC.

Theorem 1. *The algorithm* PROPERTYDRIVEN3MC$(H, \phi, l_{\mathsf{AP}}, n)$*, where the subprocedure* C3MC *implements correctly the corresponding interface, is sound w.r.t. the three-valued model checking of* ϕ *on* H*.*

5 Specializing the General Algorithmic Framework for Different Classes of Hybrid Automata

In this section we consider various classes of hybrid automata that do not admit a finite discrete bounded bisimulation. Thereof, we propose suitable specializations of our general on the fly three-valued model checking algorithm. In particular, we appropriately

instantiate the interface given in the previous section for the procedure C3MC, taking into account different possible concrete descriptions of the continuous dynamics.

5.1 The Case of Hybrid Automata Based on Autonomous ODEs and Decidable Theories over the Reals: A Symbolic Approach

The first family of hybrid automata considered is the class of autonomous hybrid automata, for which the continuous evolution is described by means of a system of autonomous ordinary differential equations (ODEs). In autonomous hybrid automata, the continuous evolution originated from a given state is the same regardless of when it starts, and thus is *uniquely* determined. An important subfamily of autonomous hybrid systems is the class of o-minimal hybrid automata [14], for which continuous evolutions are also defined within an o-minimal theory. Such an assumption guarantees the finiteness of the discrete bounded bisimulation abstraction, while its removal leads easily to the realm of infiniteness[6] [14].

To implement the interface for C3MC (cf. Figure 7), we need to consider the operators EU, AU. In virtue of the deterministic character of the continuous component in autonomous systems, we can provide suitable symbolic descriptions of the sets of states satisfying $\mathsf{E}(\varphi\mathsf{U}\psi)$, $\mathsf{A}(\varphi\mathsf{U}\psi)$. In particular, let $f = \mathsf{E}(\varphi\mathsf{U}\psi), g = \mathsf{E}(\varphi\mathsf{U}\psi)$, and suppose that you want to refine two corresponding (previously computed) sound interpretation functions $[\![f]\!]$ and $[\![g]\!]$, using the continuous evolution in a given location ℓ. Let $\Phi : \mathbb{R}_0^+ \times \mathbb{R}^n \mapsto \mathbb{R}^n$ be the flow of the system of ODE associated to ℓ. Then, the symbolic formulas (2)–(5), below, determine the sets of states within the considered location for which f and g evaluate to $v \in \{0, 1, \bot\}$, due to the continuous evolution among the regions induced by $[\![f]\!]$ and $[\![g]\!]$.

$$\iota(x,t) := \forall 0 \leq t' \leq t.\ \Phi(x,t') \in \mathit{Inv}(\ell) \tag{1}$$

$$f_{\langle 1\rangle}(x) := \exists t.\ \iota(x,t) \wedge [\![f]\!](\Phi(x,t)) = 1 \wedge \forall 0 \leq t' \leq t.\ [\![\varphi]\!](\Phi(x,t')) = 1 \tag{2}$$

$$\begin{aligned} f_{\langle \bot,1\rangle}(x) := {} & \exists t.\ \iota(x,t) \wedge ((([\![\psi]\!](\Phi(x,t)) \neq 0 \vee \\ & \exists y.\ \Phi(x,t) \xrightarrow{e} y \wedge [\![f]\!](y) \neq 0)) \wedge \forall 0 \leq t' \leq t.\ [\![\varphi]\!](\Phi(x,t')) \neq 0 \end{aligned} \tag{3}$$

$$\begin{aligned} g_{\langle 0\rangle}(x) := {} & \forall t.\ [\![\psi]\!](\Phi(x,t)) = 0 \wedge \iota(x,t) \\ & \vee \exists t.\ [\![g]\!](\Phi(x,t)) = 0 \wedge \iota(x,t) \wedge \forall 0 \leq t' < t.\ [\![\psi]\!](\Phi(x,t')) = 0 \end{aligned} \tag{4}$$

$$\begin{aligned} g_{\langle 0,\bot\rangle}(x) := {} & \forall t.\ \iota(x,t) \wedge [\![\psi]\!](\Phi(x,t)) \neq 1 \vee \forall t.\ [\![g]\!](\Phi(x,t)) = 1 \wedge \iota(x,t) \\ & \rightarrow \exists 0 \leq t' < t.\ [\![\varphi]\!](\Phi(x,t')) \neq 1 \vee \exists y.\ \Phi(x,t') \xrightarrow{e} y \wedge [\![g]\!](y) \neq 1 \end{aligned} \tag{5}$$

Let $\mathcal{A}^{dec}$ be the subclass of autonomous hybrid automata for which all relevant sets (i.e. flow, invariants, initial states, ...) are defined within a decidable theory of the reals. By Lemma 3 and 4, below, formulas (1)–(5) correctly implement SPLIT&CHECKEU and SPLIT&CHECKAU on $\mathcal{A}^{dec}$, w.r.t. the constraints imposed in the interface C3MC.

Lemma 3. *Consider the abstraction $\mathcal{A}$ of the hybrid automaton H and assume that $[\![\cdot]\!]_{\mathcal{A}}$ is sound w.r.t. the model checking of $\mathsf{E}(\phi\mathsf{U}\psi)$. Then, the following holds true:*

[6] [14] proves that autonomous (non o-minimal) *continuous* dynamical systems do not admit a finite bisimulation, implying infiniteness of 0-DBB for the corresponding hybrid systems.

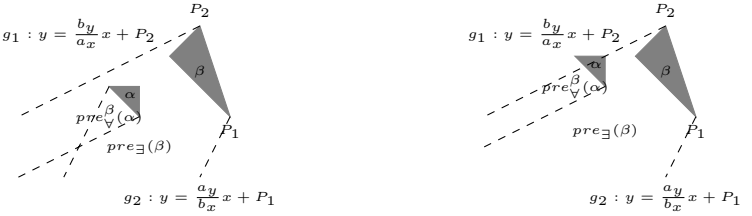


Fig. 8. Geometric approach to obtain $pre_{\exists}(\beta)$ and $pre^{\beta}_{\forall}(\alpha)$

- $x \in f_{\langle 1 \rangle} \Rightarrow [\![\mathsf{E}(\varphi \mathsf{U} \psi)]\!]_H(x) = 1$
- $[\![\mathsf{E}(\varphi \mathsf{U} \psi)]\!]_H(x) = 1 \Rightarrow x \in f_{\langle \bot, 1 \rangle}$

Lemma 4. *Consider the abstraction $\mathcal{A}$ of the hybrid automaton H and assume that $[\![\cdot]\!]_{\mathcal{A}}$ is sound w.r.t. the model checking of $\mathsf{A}(\phi \mathsf{U} \psi)$. Then, the following holds true:*

- $x \in g_{\langle 0 \rangle} \Rightarrow [\![\mathsf{A}(\varphi \mathsf{U} \psi)]\!]_H(x) = 0$
- $[\![\mathsf{A}(\varphi \mathsf{U} \psi)]\!]_H(x) = 0 \Rightarrow x \in g_{\langle 0, \bot \rangle}$

5.2 The Case of (Uninitialized) Rectangular Automata: A Geometric Approach

In this subsection we consider the class of rectangular hybrid automata (RHA) that we proved in Section 3 not to admit a finite discrete bounded bisimulation.

Unlike for hybrid automata based on autonomous ODEs, in RHA the continuous evolution traversing a state x is not uniquely determined, since it is defined by differential inclusions. In order to cope with such a non deterministic behavior for the continuous component, we introduce two basic operators $(pre_{\exists}(\alpha), pre^{\beta}_{\forall}(\alpha))$ that allow to quantify existentially (resp. universally) on the set of continuous paths evolving to the (convex) region $\alpha \subseteq \mathit{Inv}(\ell)$, within a given location ℓ. The operators $pre_{\exists}(\alpha), pre^{\beta}_{\forall}(\alpha)$ are formally defined by the Equations 6 and 7, below. Here, we use the symbol $\rightsquigarrow_{\ell}$ to represent a (feasible) continuous path within location ℓ.

$$pre_{\exists}(\alpha) := \{v \in \mathit{Inv}(\ell) \mid \exists v' \in \alpha . v \rightsquigarrow_{\ell} v'\} \tag{6}$$

$$pre^{\beta}_{\forall}(\alpha) := \{v \in pre_{\exists}(\beta) \mid \forall v' \in \beta .\ v \rightsquigarrow_{\ell} v' \Rightarrow \exists v'' \in \alpha . v \rightsquigarrow_{\ell} v'' \rightsquigarrow v'\} \tag{7}$$

Note that for the operator $pre^{\beta}_{\forall}(\alpha)$, the additional parameter β allows to restrict the universal quantification to the only paths that will eventually reach β. In particular, if β is the entire state space the operator boils down to collect the set of states for which all the paths evolve to α. The implementation of the operators $pre_{\exists}(\beta), pre^{\beta}_{\forall}(\alpha)$ on RHA can be obtained by simple geometrical reasoning. Consider e.g. the two regions α, β depicted in Figure 8, and the rectangular differential inclusion $a \leq x \leq b$. The figure illustrates the geometrical approach allowing to obtain the regions $pre_{\exists}(\beta), pre^{\beta}_{\forall}(\alpha)$. Such approach can be easily generalized to consider arbitrary located regions α, β.

Let $f = \mathsf{E}(\phi \mathsf{U} \psi)$, $g = \mathsf{A}(\phi \mathsf{U} \psi)$ for which the three-valued on the fly model checking is left open for C3MC in Figure 7. Let ℓ be a location of the hybrid automaton, let $\mathcal{P}_{\ell}$

be a partition of $Inv(\ell)$ and let $[\![f]\!]$ (resp $[\![g]\!]$) a sound three-valued interpretation of f (resp. g) over $\mathcal{P}_\ell$. Formulas (8)–(11), determine the sets of states within the location ℓ for which f and g evaluate to $v \in \{0, 1, \bot\}$, due to the continuous evolution over $\mathcal{P}_\ell$.

$$f_{\langle 1\rangle}(x) = \bigcup_{\alpha \in Inv(\ell):[\![\mathsf{E}(\phi \mathsf{U} \psi)]\!](\alpha)=1} (pre_\exists(\alpha) \setminus pre^\alpha_\forall(\bigcup_{\beta:[\![\phi]\!](\beta)\neq 1} \beta)) \tag{8}$$

$$f_{\langle \bot,1\rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell) \,:\, [\![\psi]\!](\alpha) \neq 0 \vee \\ \exists \alpha \xrightarrow{e} \gamma .\ [\![\mathsf{E}(\phi \mathsf{U} \psi)]\!](\gamma) \neq 0}} (pre_\exists(\alpha) \setminus pre^\alpha_\forall(\bigcup_{\beta:[\![\phi]\!](\beta)=0} \beta)) \tag{9}$$

$$g_{\langle 0\rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell) \,:\, [\![\mathsf{A}(\phi \mathsf{U} \psi)]\!](\alpha) = 0 \vee \\ inf(\alpha) = 1 \wedge [\![\psi]\!](\alpha) = 0)}} (pre_\exists(\alpha) \setminus pre^\alpha_\forall(\bigcup_{\beta:[\![\psi]\!](\beta)\neq 1} \beta)) \tag{10}$$

$$g_{\langle 0,\bot\rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell) \,:\, [\![\psi]\!](\alpha) \neq 1 \vee \\ \exists \alpha \xrightarrow{e} \gamma .\ [\![\mathsf{A}(\phi \mathsf{U} \psi)]\!](\gamma) \neq 1}} (pre_\exists(\alpha) \setminus pre^\alpha_\forall(\bigcup_{\beta:[\![\psi]\!](\beta)=1} \beta)) \tag{11}$$

The above sets can be used to appropriately split each class $\alpha \in \mathcal{P}_\ell$ for which $[\![f]\!](\alpha) = \bot$ (resp. $[\![g]\!](\alpha) = \bot$), refining on the fly the current information for the three-valued model checking of f (resp. g). Lemma 5 and Lemma 6, below, ensure that the sets $f_{\langle 1\rangle}(x), f_{\langle \bot,1\rangle}(x), g_{\langle 0\rangle}(x), g_{\langle 0,\bot\rangle}(x)$ are sound, and can be used to implement the functions SPLIT&CHECKEU and SPLIT&CHECKAU in procedure C3MC according to the requirements.

Lemma 5. *Consider the abstraction $\mathcal{A}$ of the hybrid automaton H and assume that $[\![\cdot]\!]_\mathcal{A}$ is sound w.r.t. the model checking of $\mathsf{E}(\phi \mathsf{U} \psi)$. Then, the following holds true:*

- $x \in f_{\langle 1\rangle} \Rightarrow [\![\mathsf{E}(\varphi \mathsf{U} \psi)]\!]_H(x) = 1$
- $[\![\mathsf{E}(\varphi \mathsf{U} \psi)]\!]_H(x) = 1 \Rightarrow x \in f_{\langle \bot,1\rangle}$

Lemma 6. *Consider the abstraction $\mathcal{A}$ of the hybrid automaton H and assume that $[\![\cdot]\!]_\mathcal{A}$ is sound w.r.t. the model checking of $\mathsf{A}(\phi \mathsf{U} \psi)$. Then, the following holds true:*

- $x \in g_{\langle 0\rangle} \Rightarrow [\![\mathsf{A}(\varphi \mathsf{U} \psi)]\!]_H(x) = 0$
- $[\![\mathsf{A}(\varphi \mathsf{U} \psi)]\!]_H(x) = 0 \Rightarrow x \in g_{\langle 0,\bot\rangle}$

6 Conclusions

We sharpen the results relative to the size of the DBB abstraction for HA, showing that the family of RHA does not admit a finite DBB. On this ground, we extend the applicability of three-valued model checking on classes of HA having infinite DBBs. In particular, we provide a property driven three-valued model checking algorithm that does not require an a-priori discretization of the states-space by means of the DBB.

References

1. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)

2. Alur, R., Henzinger, T., Ho, P.-H.: Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering 22(3), 181–201 (1996)
3. Alur, R., Henzinger, T., Lafferriere, G., Pappas, G.: Discrete abstractions of hybrid systems. Proceedings of the IEEE 88(7), 971–984 (2000)
4. Bauer, K., Gentilini, R., Schneider, K.: A uniform approach to three-valued semantics for μ-calculus on abstractions of hybrid automata. In: Hu, A., Chockler, H. (eds.) Haifa Verification Conference (HVC), Haifa, Israel. LNCS. Springer, Heidelberg (2008)
5. Brihaye, T., Michaux, C., Rivière, C., Troestler, C.: On O-minimal hybrid systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 219–233. Springer, Heidelberg (2004)
6. Fränzle, M.: What will be eventually true of polynomial hybrid automata? In: Kobayashi, N., Pierce, B.C. (eds.) TACS 2001. LNCS, vol. 2215, pp. 340–359. Springer, Heidelberg (2001)
7. Gentilini, R., Schneider, K., Mishra, B.: Successive abstractions of hybrid automata for monotonic CTL model checking. In: Artemov, S.N., Nerode, A. (eds.) LFCS 2007. LNCS, vol. 4514, pp. 224–240. Springer, Heidelberg (2007)
8. Ghosh, R., Tiwari, A., Tomlin, C.: Automated symbolic reachability analysis with application to delta-notch signaling automata. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 233–248. Springer, Heidelberg (2003)
9. Ghosh, R., Tomlin, C.: Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 232–245. Springer, Heidelberg (2001)
10. Henzinger, M., Henzinger, T., Kopke, P.: Computing simulations on finite and infinite graphs. In: Seberry, J., Pieprzyk, J. (eds.) Annual Symposium on Foundations of Computer Science (FOCS), p. 453. IEEE Computer Society Press, Los Alamitos (1995)
11. Henzinger, T.: The theory of hybrid automata. In: Verification of Digital and Hybrid Systems. NATO Advanced Study Institute Series F: Computer and Systems Sciences, vol. 170, pp. 265–292. Springer, Heidelberg (2000)
12. Henzinger, T., Kopke, P., Puri, A., Varaiya, P.: What's decidable about hybrid automata? Journal of Computer and System Sciences 57(1), 94–124 (1998)
13. Kleene, S.: Introduction to Metamathematics. North-Holland, Amsterdam (1952)
14. Lafferriere, G., Pappas, G., Sastry, S.: O-minimal hybrid systems. Mathematics of Control, Signals, and Systems 13(1), 1–21 (2000)
15. Lafferriere, G., Pappas, J., Yovine, S.: A new class of decidable hybrid systems. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 137–151. Springer, Heidelberg (1999)
16. Miller, J.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 296–309. Springer, Heidelberg (2000)
17. Piazza, C., Antoniotti, M., Mysore, V., Policriti, A., Winkler, F., Mishra, B.: Algorithmic algebraic model checking I: Challenges from systems biology. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 5–19. Springer, Heidelberg (2005)
18. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation based abstraction refinement. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 573–589. Springer, Heidelberg (2005)
19. Tiwari, A., Khanna, G.: Series of abstractions for hybrid automata. In: Tomlin, C.J., Greenstreet, M.R. (eds.) HSCC 2002. LNCS, vol. 2289, pp. 465–478. Springer, Heidelberg (2002)

Team Logic and Second-Order Logic*

Juha Kontinen and Ville Nurmi

Department of Mathematics and Statistics, P.O. Box 68, 00014 University of Helsinki, Finland
{juha.kontinen,ville.v.nurmi}@helsinki.fi

Abstract. Team logic is a new logic, introduced by Väänänen [1], extending dependence logic by classical negation. Dependence logic adds to first-order logic atomic formulas expressing functional dependence of variables on each other. It is known that on the level of sentences dependence logic and team logic are equivalent with existential second-order logic and full second-order logic, respectively. In this article we show that, in a sense that we make explicit, team logic and second-order logic are also equivalent with respect to open formulas. A similar earlier result relating open formulas of dependence logic to the negative fragment of existential second-order logic was proved in [2].

1 Introduction

Team logic is a new logic arising from the so-called Dependence Logic [1] by adding classical negation. Dependence logic adds the concept of dependence to first-order logic by means of new atomic formulas

$$=(x_1, \ldots, x_n, y) \tag{1}$$

the meaning of which is that the values of $x_1, \ldots, x_n$ completely determine the values of y. This is made precise in dependence logic by basing semantics in the concept of a *set* of assignments satisfying a formula (rather than an individual assignment satisfying a formula, as in first-order logic). Such sets are called *teams*. The idea of team based semantics is due to Hodges [3]. A team X is said to satisfy the formula (1) if any two assignments s and s' from X that agree about $x_1, \ldots, x_n$ also agree about y. If we think of X as a database, this is the concept of *functional dependence*, studied extensively in database theory.

We review briefly the short history of team logic. In first-order logic the order in which quantifiers are written determines the mutual dependence relations between variables. For example, in

$$\forall x_0 \exists x_1 \forall x_2 \exists x_3 \phi$$

* The first author was supported by grant 127661 of the Academy of Finland and the European Science Foundation Eurocores programme LogICCC [FP002 - Logic for Interaction (LINT)] through grant 129208 of the Academy of Finland. The second author was supported by the MALJA Graduate school in Mathematical logic.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 230–241, 2009.

the variable x_1 depends on x_0, and the variable x_3 depends on both x_0 and x_2. In dependence logic we write down explicitly the dependence relations between variables and by so doing make it possible to express dependencies not otherwise expressible in first-order logic.

The first step in this direction was taken by Henkin [4] with his partially ordered quantifiers, e.g.

$$\begin{pmatrix} \forall x_0 \; \exists x_1 \\ \forall x_2 \; \exists x_3 \end{pmatrix} \phi,$$

where x_1 depends only on x_0 and x_3 depends only on x_2. The second step was taken by Hintikka and Sandu [5,6], who introduced the slash-notation

$$\forall x_0 \exists x_1 \forall x_2 \exists x_3 / \forall x_0 \phi,$$

where $\exists x_3 / \forall x_0$ means that x_3 is "independent" of x_0 in the sense that a choice for the value of x_3 should not depend on what the value of x_0 is. The observation of Hintikka and Sandu was that we can add slashed quantifiers $\exists x_3 / \forall x_0$ coherently to first-order logic if we give up some of the classical properties of negation, most notably the Law of Excluded Middle. They called their logic Independence Friendly Logic (IF). In [1] Väänänen takes the further step of writing down explicitly the mutual dependence relationships between variables. Thus he writes

$$\forall x_0 \exists x_1 \forall x_2 \exists x_3 (=(x_2, x_3) \wedge \phi) \tag{2}$$

to indicate that x_3 depends on x_2 only. The new atomic formula $=(x_2, x_3)$ has the explicit meaning that x_3 depends on x_2 and on nothing else. This results in a logic Väänänen calls Dependence Logic. It is equivalent in expressive power to the logic of Hintikka and Sandu in the sense that there are truth-preserving translations from one to the other. Both have the same expressive power on the level of sentences being equivalent to the existential fragment of second-order logic.

The negation $\neg$ of dependence logic does not satisfy the law of excluded middle and is therefore not the classical Boolean negation. This is clearly manifested by the existence of non-determined sentences ϕ in dependence logic. In such cases, the failure of $\mathcal{M} \models \phi$ does not imply $\mathcal{M} \models \neg\phi$. Hintikka [6] introduced extended independence friendly logic by taking the Boolean closure of his independence friendly logic. We take the further action of making classical negation $\sim$ one of the logical operations on a par with other propositional operations and quantifiers. This yields an extension of the Boolean closure of dependence logic. The new logic is called Team Logic in [1].

While we define team logic we have to restrict the negation $\neg$ of dependence logic. The game-theoretic intuition behind $\neg\phi$ is that it says something about "the other player." The introduction of $\sim$ unfortunately ruins the basic game-theoretic intuition, and there is no "other player" anymore. If ϕ is a formula of dependence logic, then $\sim\phi$ has the meaning "player **II** does not have a winning strategy," but it is not clear what the meaning of $\neg\sim\phi$ would be.

Just as a formula ϕ of first-order logic with free variables $x_1, \ldots, x_n$ defines an n-ary relation in a model $\mathcal{M}$, i.e., the set of n-tuples $(a_1, \ldots, a_n)$ that satisfy

ϕ in $\mathcal{M}$, a formula ϕ of team logic defines a set of n-ary relations, namely the set of teams that satisfies ϕ. It was shown in [2] that a set $\mathcal{X}$ of n-ary relations is defined in this sense by a formula $\phi(x_1, \ldots, x_n)$ of dependence logic if and only if $\mathcal{X}$ is definable in existential second-order logic by a formula $\Phi(P)$ with an n-ary predicate symbol P occuring negatively only. The goal of this paper is to prove a similar result for team logic, namely, we show that a set $\mathcal{X}$ of n-ary relations is defined in this sense by a formula $\phi(x_1, \ldots, x_n)$ of team logic if and only if $\mathcal{X}$ is definable in second-order logic by a formula $\Phi(P)$ with an n-ary predicate symbol P.

2 Preliminaries

In this section we define dependence logic and team logic and recall some of their properties.

The syntax of dependence logic and team logic extend the syntax of first-order logic by new atomic (dependence) formulas of the form

$$=(t_1, \ldots, t_n), \tag{3}$$

where $t_1, \ldots, t_n$ are terms. The intuitive meaning of the dependence formula (3) is that the value of the term t_n is determined by the values of the terms $t_1, \ldots, t_{n-1}$. As singular cases we have

$$=(),$$

which we take to be universally true, and

$$=(t),$$

which declares that the value of the term t depends on nothing, i.e., is constant.

In order to define the semantics of dependence logic and team logic, we first need to define the concept of a *team*. Let $\mathcal{M}$ be a model with domain M. *Assignments* of $\mathcal{M}$ are finite mappings from variables into M. The value of a term t in an assignment s is denoted by $\langle t \rangle^s$ or $\langle t \rangle^{\overline{f},s}$ if t contains some function variables whose interpretations are given by $\overline{f}$.

If s is an assignment, x a variable, and $a \in M$, then $s(x \mapsto a)$ denotes the assignment which agrees with s everywhere except that it maps x to a.

Let M be a set and $\{x_1, \ldots, x_k\}$ a finite (possibly empty) set of variables. A *team* X of M with domain $\{x_1, \ldots, x_k\}$ is any set of assignments from the variables $\{x_1, \ldots, x_k\}$ into the set M. In particular, there are two teams with the empty domain, namely, the empty team $\emptyset$ and the full team $\{\emptyset\}$. We denote by $\mathrm{Rel}(X)$ the k-ary relation of M corresponding to X

$$\mathrm{Rel}(X) = \{(s(x_1), \ldots, s(x_k)) : s \in X\} \ .$$

Definition 1. *Let X be a team of M, $a \in M$ and $F\colon X \to M$. The following operations on teams will be used:*

$$X(x \mapsto a) := \{s(x \mapsto a) : s \in X\}$$
$$X(x \mapsto F) := \{s(x \mapsto F(s)) : s \in X\}$$
$$X(x \mapsto M) := \{s(x \mapsto a) : s \in X \text{ and } a \in M\}$$

It might help one's intuition to note that the first two operations can only decrease the cardinality of a team, whereas the third operation can potentially multiply team's cardinality by the factor of $|M|$.

We are now ready to define the syntax and semantics of dependence logic and team logic. Since team logic is the closure of dependence logic under classical negation, we will first give simultaneously the syntax and semantics of team logic and then indicate how dependence logic can be acquired as a fragment of team logic.

Definition 2. *Let L be a vocabulary, $\mathcal{M}$ a L-model and X a team of M. We define the satisfaction relation $\mathcal{M}, X \models \phi$ for L-formulas of team logic in the following way. Below, we assume that the domain of X contains at least the variables which appear free in ϕ. Also, the case of atomic formulas refers to the usual truth definition of first-order logic.*

$$\begin{aligned}
&\mathcal{M}, X \models Rt_1, \ldots, t_k && \iff \text{for all } s \in X\colon \mathcal{M}, s \models Rt_1, \ldots, t_k\\
&\mathcal{M}, X \models \neg Rt_1, \ldots, t_k && \iff \text{for all } s \in X\colon \mathcal{M}, s \models \neg Rt_1, \ldots, t_k\\
&\mathcal{M}, X \models =(t_1, \ldots, t_k, u) && \iff \text{there is } f \text{ such that for all } s \in X\colon\\
&&& \quad \langle u \rangle^s = f(\langle t_1 \rangle^s, \ldots, \langle t_k \rangle^s)\\
&\mathcal{M}, X \models \neg=(t_1, \ldots, t_k, u) && \iff X = \emptyset\\
&\mathcal{M}, X \models \sim\phi && \iff \mathcal{M}, X \not\models \phi\\
&\mathcal{M}, X \models \phi \vee \psi && \iff \mathcal{M}, X \models \phi \text{ or } \mathcal{M}, X \models \psi\\
&\mathcal{M}, X \models \phi \wedge \psi && \iff \mathcal{M}, X \models \phi \text{ and } \mathcal{M}, X \models \psi\\
&\mathcal{M}, X \models \phi \otimes \psi && \iff \text{there is } Y, Z \subseteq X \text{ such that } Y \cup Z = X \text{ and}\\
&&& \quad \mathcal{M}, Y \models \phi \text{ and } \mathcal{M}, Z \models \psi\\
&\mathcal{M}, X \models \phi \oplus \psi && \iff \text{for all } Y, Z \subseteq X\colon \text{if } Y \cup Z = X \text{ then}\\
&&& \quad \mathcal{M}, Y \models \phi \text{ or } \mathcal{M}, Z \models \psi\\
&\mathcal{M}, X \models \exists x\phi && \iff \text{there is } F\colon X \to M \text{ such that}\\
&&& \quad \mathcal{M}, X(x \mapsto F) \models \phi\\
&\mathcal{M}, X \models \forall x\phi && \iff \text{for all } F\colon X \to M\colon \mathcal{M}, X(x \mapsto F) \models \phi\\
&\mathcal{M}, X \models !x\phi && \iff \mathcal{M}, X(x \mapsto M) \models \phi
\end{aligned}$$

Finally, a sentence ϕ is true in a model $\mathcal{M}$ if $\mathcal{M}, \{\emptyset\} \models \phi$.

As already mentioned, team logic is the extension of dependence logic by classical negation $\sim$. Dependence logic can be acquired as the fragment of team logic

allowing only atomic formulas and their negations ($\neg$) and the connectives $\wedge$, $\otimes$, $\exists$ and !. In the context of dependence logic, the connectives $\otimes$ and ! are usually denoted by $\vee$ and $\forall$. However, in team logic classical disjunction $\vee$ and the connectives $\oplus$ and $\forall$ arise as the duals of $\wedge$, $\otimes$ and $\exists$ with respect to $\sim$. Therefore, in this paper we think of dependence logic as the fragment of team logic using only the connectives $\wedge$, $\otimes$, $\exists$ and !.

It is instructive to note that also first-order logic can be embedded into team logic (and dependence logic) in a canonical way. The canonical translation of a first-order formula ϕ (in negation normal-form) into a team logic formula ϕ^* goes as follows.

$$\begin{aligned} (Rt_1,\dots,t_k)^* &:= Rt_1,\dots,t_k & (\neg\phi)^* &:= \neg\phi^* \\ (\phi\vee\psi)^* &:= \phi^*\otimes\psi^* & (\phi\wedge\psi)^* &:= \phi^*\wedge\psi^* \\ (\exists x\phi)^* &:= \exists x\phi^* & (\forall x\phi)^* &:= !x\phi^* \end{aligned}$$

It is easy to verify that this translation satisfies

$$\mathcal{M},X\models\phi^* \iff \forall s\in X:\ \mathcal{M},s\models\phi,$$

from which it follows, for any sentence ϕ, that ϕ and ϕ^* are logically equivalent.

3 Background

In this section we discuss the expressive power of dependence logic and team logic and formalize the goal of this paper. We begin with dependence logic.

It is known that, on the level of sentences, the expressive power of dependence logic coincides with that of existential second-order logic (Σ^1_1):

Theorem 1. *For every sentence ϕ of dependence logic there is a sentence Φ of Σ^1_1 such that*

$$\text{For all models } \mathcal{M}\text{: } \mathcal{M}\models_{\{\emptyset\}}\phi \iff \mathcal{M}\models\Phi\ . \tag{4}$$

Conversely, for every sentence Φ of Σ^1_1 there is a sentence ϕ of dependence logic such that (4) holds.

Proof. Using the method of [7,8] (See Theorems 6.2 and 6.15 in [1]).

Theorem 1 does not tell us, on the face of it, anything about the formulas of dependence logic with free variables. Recall that a formula ϕ of first-order logic, with free variables $x_1,\dots,x_n$, defines an n-ary relation in a model $\mathcal{M}$, i.e., the set of n-tuples $(a_1,\dots,a_n)$ that satisfy ϕ in $\mathcal{M}$. On the other hand, a formula $\phi(x_1,\dots,x_n)$ of team logic defines, in a model $\mathcal{M}$, a set of n-ary relations $\mathrm{Rel}(X)\subseteq M^n$, namely, the set of teams X with domain $\{x_1,\dots,x_n\}$ that satisfy ϕ in $\mathcal{M}$. It was proved in [2] that a set $\mathcal{X}$ of n-ary relations is defined in this sense by a formula $\phi(x_1,\dots,x_n)$ of dependence logic if and only if $\mathcal{X}$ is definable in existential second-order logic by a sentence $\Phi(P)$ with an n-ary predicate symbol

P occuring negatively only. On the semantical side, the assumption that P can only occur negatively corresponds to downward monotonicity. The following fact (Fact 11.1 in [3], see Proposition 3.10 in [1]) implies that the result in [2] is the best possible:

Proposition 1 (Downward closure). *Suppose $Y \subseteq X$. Then $\mathcal{M}, X \models \varphi$ implies $\mathcal{M}, Y \models \varphi$.*

Our goal in this article is to characterize definable sets of teams, i.e., sets of the form,

$$\{X : \mathcal{M}, X \models \phi\}, \tag{5}$$

in team logic. It was shown in [9] that on the level of sentences the expressive power of team logic corresponds to full second-order logic (SO).

Theorem 2. *For every sentence ϕ of team logic there is a sentence Φ of second-order logic such that*

$$\text{For all models } \mathcal{M}\text{: } \mathcal{M} \models_{\{\emptyset\}} \phi \iff \mathcal{M} \models \Phi \ . \tag{6}$$

Conversely, for every sentence Φ of second-order logic there is a sentence ϕ of team logic such that (6) holds.

In this article our goal is to generalize Theorem 2 from sentences to arbitrary formulas.

4 Examples

In this section we illustrate the question studied in this paper with examples.

Example 1. Let L be a vocabulary, $\mathcal{M}$ a L-model, and $\mathcal{X}$ a family of sets of r-tuples of M. In [2] it was proved that the following are equivalent:

1. $\mathcal{X} = \{\mathrm{Rel}(X) : \mathcal{M}, X \models \psi(v_1, \ldots, v_r)\}$ for some L-formula $\psi(v_1, \ldots, v_r)$ of dependence logic.
2. $\mathcal{X} = \{Y : \mathcal{M}, Y \models \phi(R)\}$ for some sentence $\phi \in \Sigma^1_1[L \cup \{R\}]$, in which R occurs only negatively.

Since team logic is closed under boolean operations, it follows that the family of definable collections $\mathcal{X}$ of teams over $\mathcal{M}$ is closed under boolean operations and thus properly contains the family of $\mathcal{X}$ which can be defined in dependence logic.

In the following example we show that very simple formulas can be used to express properties of teams which cannot be expressed by any formula of dependence logic.

Example 2. Let $\phi(x)$ and $\psi(x, y)$ be the formulas

$$\psi(x, y) := {\sim}{=}(x, y)$$
$$\phi(x) := {\sim}\neg{=}(x).$$

Let $\mathcal{M}$ be a model and X is a team of M with domain $\{x, y\}$. It is easy to verify that

$$\mathcal{M}, X \models \psi(x, y) \text{ iff } \exists s, s' \in X(s(x) = s'(x) \wedge s(y) \neq s'(y)) \ .$$

On the other hand, it is immediate that for all $\mathcal{M}$ and X:

$$\mathcal{M}, X \models \phi(x) \text{ iff } X \neq \emptyset \ .$$

Note that the formula $\psi(x, y)$ does not satisfy the downward closure as defined in Proposition 1 and therefore it cannot be expressed in dependence logic. On the other hand, the formula $\phi(x)$ is satisfied by all non-empty teams and hence it cannot be expressed in dependence logic since all formulas of dependence logic are satisfied by the empty team [1].

The next example shows that team logic is closed under interesting operations which can be used to define new properties of teams from any property which is already definable (see Examples 8.14 and 8.15 in [1]).

Example 3. Suppose that $\psi(\overline{x})$ is a formula of team logic. There are formulas $\varphi(\overline{x})$ and $\theta(\overline{x})$ of team logic such that for all models $\mathcal{M}$ and teams X:

$$\mathcal{M}, X \models \varphi(\overline{x}) \iff \forall Y \subseteq X(\mathcal{M}, Y \models \psi(\overline{x}))$$
$$\mathcal{M}, X \models \theta(\overline{x}) \iff \forall Y \subseteq X \exists Z \subseteq Y(\mathcal{M}, Z \models \psi(\overline{x})).$$

In our last example we indicate that team logic has interesting applications also outside of logic.

Example 4. Recall that *Arrow's theorem* in social choice theory shows that no voting system, with three or more discrete options to choose from, can convert the ranked preferences of individuals into a community-wide ranking which would meet a certain set of reasonable criteria. Equivalently, if a social welfare function satisfies a certain set of reasonable criteria with at least three discrete options to choose from then it is a *dictatorship*, i.e., there is a single person whose preferences determine the preferences of the social welfare function. Arrow's Theorem can be interpreted as a valid sentence in team logic [10]. Without going to the details we note the logical form of the corresponding sentence is

$$!\overline{x}\forall\overline{y}(\sim(\phi \wedge =(\overline{t}, v)) \vee (\bigvee_{i=1}^{N}(t_i = v))),$$

where ϕ is a quantifier-free first-order formula.

5 The Main Result

In this section we characterize the properties of teams definable in team logic. In other words, we characterize the expressive power of open formulas of team logic.

The following theorem (Theorem 8.13 in [1]) gives an upper-bound for the solution.

Theorem 3. *Let L be a vocabulary and $\phi(v_1, \ldots, v_r)$ a L-formula of team logic with free variables $v_1, \ldots, v_r$. Then there is a sentence $\eta_\phi(R)$, where R is r-ary, of second-order logic such that for all model $\mathcal{M}$ and teams X with domain $\{v_1, \ldots, v_r\}$:*

$$\mathcal{M}, X \models \phi \iff \mathcal{M}, \mathrm{Rel}(X) \models \eta_\phi(R) \ .$$

We will next show that the converse of Theorem 3 also holds. In our proof we will be using the fact that formulas of second-order logic can be transformed to the so-called Skolem Normal Form [11] (see [12]).

Lemma 1. *Every formula of second-order logic is equivalent to a formula of the form*

$$\exists f_1^1 \ldots \exists f_n^1 \, \forall f_1^2 \ldots \forall f_n^2 \ \ldots \ \exists f_1^p \ldots \exists f_n^p \, \forall x_1 \ldots \forall x_n \, \theta,$$

where θ is a quantifier-free formula.

Theorem 4. *Let L be a vocabulary, R r-ary predicate symbol not in L, and $\phi(R)$ an $L \cup \{R\}$-sentence of second-order logic. Then there is an L-formula of team logic $\psi(v_1, \ldots, v_r)$ such that for all models $\mathcal{M}$ and teams X with domain $\{v_1, \ldots, v_r\}$:*

$$\mathcal{M}, X \models \psi \iff \mathcal{M}, \mathrm{Rel}(X) \models \phi \ .$$

Proof. See the Appendix. □

Theorem 4 can be also localized to a fixed model analogously to Theorem 4.10 in [2].

Corollary 1. *Let L be a vocabulary, $\mathcal{M}$ a L-model, and $\mathcal{X}$ a family of sets of r-tuples of M. Then the following are equivalent:*

1. *$\mathcal{X} = \{\mathrm{Rel}(X) : \mathcal{M}, X \models \psi(v_1, \ldots, v_r)\}$ for some L-formula $\psi(v_1, \ldots, v_r)$ of team logic.*
2. *$\mathcal{X} = \{Y : \mathcal{M}, Y \models \phi(R)\}$ for some sentence $\phi \in \mathrm{SO}[L \cup \{R\}]$.*

6 Applications

In this section we discuss applications of Theorem 4 and present an open problem.

Theorem 4 shows that team logic is equivalent to full second-order logic in a strong way. In [2] an analogous semantical "completeness" result of dependence logic with repect to (negative) Σ_1^1 was used to argue that extending dependence logic by certain new connectives does not increase the expressive power of dependence logic. Theorem 4 allows us to reason analogously about team logic. The motivation for this line of study arises from the fact that there are many natural and interesting connectives that can be used with logics having semantics based on teams (see, e.g., [13]). In this respect Theorem 4 also shows that the set of connectives we used to define team logic gives rise to the (semantically) maximal

logic which is equivalent to second-order logic both on the level of sentences and open formulas.

We will next take an example of a new connective $\hookrightarrow$ that could be added to team logic. Define the connective $\hookrightarrow$ by the clause:

$$\mathcal{M}, X \models \phi \hookrightarrow \psi \iff \text{for all } Y \subseteq X, \text{ if } \mathcal{M}, Y \models \phi \text{ and for all } Z \text{ such that } Y \subsetneq Z \subseteq X \text{ it holds that } \mathcal{M}, Z \not\models \phi, \text{ then } \mathcal{M}, Y \models \psi.$$

Denote by TL($\hookrightarrow$) the extension of team logic by $\hookrightarrow$. It is straightforward to verify that the connective $\hookrightarrow$ can be expressed in second-order logic. In particular, the proof of Theorem 3 can be extended to cover also the case of $\hookrightarrow$. Now, by Theorem 4 we can actually translate any formula of the logic TL($\hookrightarrow$) into an equivalent team logic formula. The question remaining open is whether there is a compositional way of doing this translation which does not use the translation to second-order logic and Theorem 4.

Acknowledgments. We are grateful to Jouko Väänänen for valuable comments and suggestions in all the stages of writing this article.

References

1. Väänänen, J.: Dependence logic: A New Approach to Independence Friendly Logic. London Mathematical Society Student Texts, vol. 70, p. 234. Cambridge University Press, Cambridge (2007)
2. Kontinen, J., Väänänen, J.: On definability in dependence logic. To appear in Journal of Logic, Language and Information (2007)
3. Hodges, W.: Compositional semantics for a language of imperfect information. Log. J. IGPL 5(4), 539–563 (1997) (electronic)
4. Henkin, L.: Some remarks on infinitely long formulas. In: Infinitistic Methods (Proc. Sympos. Foundations of Math., Warsaw, 1959), pp. 167–183. Pergamon Press, Oxford (1961)
5. Hintikka, J., Sandu, G.: Informational independence as a semantical phenomenon. In: Logic, methodology and philosophy of science, VIII (Moscow, 1987). Stud. Logic Found. Math, vol. 126, pp. 571–589. North-Holland, Amsterdam (1989)
6. Hintikka, J.: The Principles of Mathematics Revisited. Cambridge University Press, Cambridge (1996)
7. Walkoe Jr., W.J.: Finite partially-ordered quantification. J. Symbolic Logic 35, 535–555 (1970)
8. Enderton, H.B.: Finite partially-ordered quantifiers. Z. Math. Logik Grundlagen Math. 16, 393–397 (1970)
9. Nurmi, V.: Dependence Logic: Investigations into Higher-Order Semantics Defined on Teams. PhD thesis, University of Helsinki (2009)
10. Väänänen, J.: Personal communication (2009)
11. Skolem, T.: Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze nebst einem Theoreme über dichte Mengen. Skrifter utgit av Videnskappsselskapet i Kristiania (1920)
12. Skolem, T.: Selected works in logic. Edited by Jens Erik Fenstad. Universitetsforlaget, Oslo (1970)

13. Abramsky, S., Väänänen, J.: From IF to BI, a tale of dependence and separation. Technical report, University of Amsterdam, ILLC Prepublication Series, PP-2008-27 (2008)

Appendix

Proof (of Theorem 4). By Lemma 1, we may assume that ϕ is of the form

$$\exists f_1^1 \dots \exists f_n^1 \, \forall f_1^2 \dots \forall f_n^2 \; \dots \; \exists f_1^p \dots \exists f_n^p \, \forall x_1 \dots \forall x_n \; \theta', \tag{7}$$

where θ' is quantifier-free. We may further assume (see the proof of Theorem 6.15 in [1] for details) that θ' is in conjunctive normal form, all occurences of the relation variable R in θ' are of the form $R\overline{x}$, where $\overline{x}$ denotes the sequence $x_1, \dots, x_r$, and finally each function variable f_j^i occurs in θ' only in occurrences of the term $t_j^i := f_j^i u_1^{i,j}, \dots, u_{k(i,j)}^{i,j}$, where each $u_k^{i,j}$ is a variable. We can rewrite ϕ in the logically equivalent form

$$\phi := \mathop{\exists}_{j \le n} f_j^1 \mathop{\forall}_{j \le n} f_j^2 \; \dots \mathop{\exists}_{j \le n} f_j^p \mathop{\forall}_{j \le n} x_j \; \theta, \tag{8}$$

where n is possibly increased for the sake of obtaining two new function variables f_{n-1}^p and f_n^p which we shall simply call f_1 and f_2, and

$$\theta := (R\overline{x} \vee \neg(f_1\overline{x} = f_2\overline{x})) \wedge (\neg R\overline{x} \vee f_1\overline{x} = f_2\overline{x}) \wedge \theta',$$

where we replace the occurrences of $R\overline{x}$ in θ' by $f_1\overline{x} = f_2\overline{x}$. Note that θ is in conjunctive normal form just like θ' and there are only one positive and one negative occurrence of $R\overline{x}$ in θ, both occurrences in their own conjuncts.

Let ψ be the team logic formula

$$\psi := \mathop{!}_{i \le n} x_i \mathop{\exists}_{j \le n} y_j^1 \mathop{\forall}_{j \le n} y_j^2 \; \dots \mathop{\exists}_{j \le n} y_j^p \Big(\sim \bigwedge_{\substack{i \le p \text{ even} \\ j \le n}} \chi_j^i \vee \Big(\bigwedge_{\substack{i \le p \text{ odd} \\ j \le n}} \chi_j^i \wedge \theta^* \Big) \Big),$$

where θ^* is the canonical translation of θ into team logic (defined in Sect. 2) with the following replacements of terms and subformulas;

$$\begin{aligned} t_j^i &\mapsto y_j^i \\ \neg R\overline{x} &\mapsto \bigotimes_{i \le r} \neg(v_i = x_i) \\ R\overline{x} &\mapsto \mathop{\forall}_{i \le r} z_i \Big(\bigvee_{i \le r} {\sim}{=}(z_i) \vee {\sim} \bigotimes_{i \le r} \neg(v_i = z_i) \vee \bigotimes_{i \le r} \neg(x_i = z_i) \Big), \end{aligned}$$

where v_i and z_i, for $i \le r$, are new variables, and χ_j^i for $i \le p$ and $j \le n$ we define as

$$\chi_j^i := {=}(u_1^{i,j}, \dots, u_{k(i,j)}^{i,j}, y_j^i) \; .$$

We shall call y^p_{n-1} simply y_1 and y^p_n we call y_2, according to the notation of f_1 and f_2.

Let X be a team with domain $\{v_1, \dots, v_r\}$. Assume first that $\mathcal{M}, \mathrm{Rel}(X) \models \phi$. We will show $\mathcal{M}, X \models \psi$. From $\mathcal{M}, \mathrm{Rel}(X) \models \phi$ we get that alternately for each odd $i \leq p$ we can pick some particular sequence $f^i_1, \dots, f^i_n$ of functions such that whichever sequence we pick for each even $i \leq p$, it always results in $\mathcal{M}, \mathrm{Rel}(X), (f^i_j)_{i,j} \models \forall x_1 \dots \forall x_n \theta$. We can translate this same function picking strategy to the side of team logic; for each i and j, let F^i_j map assignments like $F^i_j(s) = f^i_j\big(s(u^{i,j}_1), \dots, s(u^{i,j}_{k(i,j)})\big)$. If we can show that team Y, where

$$Y := X(x_i \mapsto M)_{i \leq n}(y^i_j \mapsto F^i_j)_{i \leq p,\ j \leq n}, \tag{9}$$

satisfies the conditional subformula of ψ, we get that $\mathcal{M}, X \models \psi$. The team Y satisfies χ^i_j for each i and j. Therefore, in order for X to satisfy ψ, Y should satisfy θ^*, i.e. Y should satisfy each of the conjuncts in θ^*. Note that, for each $s \in Y$, $\langle y^i_j \rangle^s = \langle t^i_j \rangle^{f^i_j, s}$.

Note that in essence we are proving the powerful claim that, for arbitrary interpretations of the function variables f^i_j for all i and j, it holds that

$$\mathcal{M}, \mathrm{Rel}(X), (f^i_j)_{i,j} \models \forall x_1 \dots \forall x_n \theta \iff \mathcal{M}, Y \models \theta^*, \tag{10}$$

where Y extends X as defined in (9).

There are three types of conjuncts in θ^*. We will show that Y satisfies conjuncts of all the three types. This is the implication from left to right in (10).

- To see that Y satisfies the formula that replaced $R\overline{x} \vee \neg(f_1\overline{x} = f_2\overline{x})$, consider any $s \in Y$. From $\mathcal{M}, \mathrm{Rel}(X), (f^i_j)_{i,j}, s \models \theta$ we get either $\mathcal{M}, \mathrm{Rel}(X), s \models R\overline{x}$ or $\mathcal{M}, (f^i_j)_{i,j}, s \models \neg(f_1\overline{x} = f_2\overline{x})$. We can then split $Y = Y_1 \cup Y_2$, where $Y_1 = \{s \in Y : (s(x_1), \dots, s(x_r)) \in \mathrm{Rel}(X)\}$ and $Y_2 = Y \setminus Y_1$ such that $s(y_1) \neq s(y_2)$ holds for all $s \in Y_2$. Thus $\mathcal{M}, Y_2 \models \neg(y_1 = y_2)$. We also have that, for all $s \in Y_1$ and all $(a_1, \dots, a_r) \in \mathrm{Rel}(X)$ there is some $s' \in Y_1$ such that $s'(v_i) = a_i$ for all $i \leq r$ and $s'(x_i) = s(x_i)$ for all $i \leq n$.
  Let $(F_1, \dots, F_r)$ be a sequence of successive supplement functions for Y_1 and consider $Z := Y_1(z_i \mapsto F_i)_{i \leq r}$. If some F_i is not a constant function, then $\mathcal{M}, Z \models {\sim}{=}(z_i)$ and thus $\mathcal{M}, Z \models \bigvee_{i \leq r} {\sim}{=}(z_i)$. Otherwise $Z = Y_1(z_i \mapsto a_i)_{i \leq r}$ for some $(a_1, \dots, a_r) \in M^r$. If for all $s \in Z$ there is $i \leq r$ such that $s(x_i) \neq a_i$, then $\mathcal{M}, Z \models \bigotimes_{i \leq r} \neg(x_i = z_i)$. Otherwise there is $s \in Z$ such that $s(x_i) = a_i$ for all $i \leq r$. Then there is some $s' \in Z$, where $s'(v_i) = s(x_i)$ for all $i \leq r$. Since we must have $s'(z_i) = s(z_i) = a_i$, for all $i \leq r$, it holds that $s'(v_i) = s(x_i) = a_i = s'(z_i)$ for all $i \leq r$, whence $\mathcal{M}, Z \models {\sim} \bigotimes_{i \leq r} \neg(v_i = z_i)$. This shows that Y satisfies the conjunct.
- To see $\mathcal{M}, Y \models \bigotimes_{i \leq r} \neg(v_i = x_i) \otimes y_1 = y_2$, consider any $s \in Y$. As above, we get from $\mathcal{M}, \mathrm{Rel}(X), (f^i_j)_{i,j}, s \models \theta$ either $\mathcal{M}, \mathrm{Rel}(X), s \models \neg R\overline{x}$ or $\mathcal{M}, (f^i_j)_{i,j}, s \models f_1\overline{x} = f_2\overline{x}$. We can then split $Y = Y_1 \cup Y_2$, where $Y_1 = \{s \in Y : (s(x_1), \dots, s(x_r)) \notin \mathrm{Rel}(X)\}$ and $Y_2 = \{s \in Y : s(y_1) = s(y_2)\}$. Then $\mathcal{M}, Y_1 \models \bigotimes_{i \leq r} \neg(v_i = x_i)$ because for each $s \in Y_1$, $(s(v_1), \dots, s(v_r)) \in \mathrm{Rel}(X)$. Clearly $\mathcal{M}, Y_2 \models y_1 = y_2$.

– To see $\mathcal{M}, Y \models \bigotimes_{i\leq q} \alpha_i^*$, where each α_i is an atomic formula where R does not occur, simply split $Y = \bigcup_{i\leq q} Y_i$ such that each $Y_i = \{s \in Y : \mathcal{M}, (f_j^i)_{i,j}, s \models \alpha_i\}$. Then $\mathcal{M}, Y_i \models \alpha_i^*$ for each $i \leq q$.

For the other direction, assume $\mathcal{M}, X \models \psi$. We will show $\mathcal{M}, \mathrm{Rel}(X) \models \phi$. From $\mathcal{M}, X \models \psi$ we get $\mathcal{M}, Y \models \theta^*$, where Y is as in (9) for certain sequences of supplement functions F_j^i. We translate them into functions f_j^i by setting for each sequence of $a_1, \ldots, a_{k(i,j)} \in M$, $f_j^i(a_1, \ldots, a_{k(i,j)}) = F_j^i(s)$, where $s(u_k^{i,j}) = a_k$ for each $k \leq k(i,j)$. Then each f_j^i is well defined because $\mathcal{M}, Y \models \chi_j^i$ for each i and j. If we can show that $\mathcal{M}, \mathrm{Rel}(X), (f_j^i)_{i,j} \models \forall x_1 \ldots \forall x_n \theta$, we get $\mathcal{M}, \mathrm{Rel}(X) \models \phi$. To this end, let s be an assignment that interprets each x_i, $i \leq n$. There are (possibly several) extensions of s in Y that we will be referring to. To make things simple, we can think that s itself is one of those extensions in Y—the only difference is that s then interprets the additional variables $v_1, \ldots, v_r, y_1^1, \ldots, y_n^p$ that do not occur in θ. Note that $\langle y_j^i \rangle^s = \langle t_j^i \rangle^{f_j^i, s}$.

We will next show the implication from right to left in (10), i.e. we will show it for the three kinds of conjuncts in θ for an arbitrary s.

– To see $\mathcal{M}, \mathrm{Rel}(X), (f_j^i)_{i,j}, s \models R\overline{x} \vee \neg(f_1\overline{x} = f_2\overline{x})$, assume $\mathcal{M}, (f_j^i)_{i,j}, s \models f_1\overline{x} = f_2\overline{x}$. Then note that from $\mathcal{M}, Y \models \theta^*$ we get a split $Y = Y_1 \cup Y_2$ such that

$$\mathcal{M}, Y_1 \models \mathop{\mathcal{A}}_{i\leq r} z_i \Big(\bigvee_{i\leq r} {\sim}{=}(z_i) \vee {\sim} \bigotimes_{i\leq r} \neg(v_i = z_i) \vee \bigotimes_{i\leq r} \neg(x_i = z_i) \Big) \tag{11}$$

and $\mathcal{M}, Y_2 \models \neg(y_1 = y_2)$. Because $s(y_1) = s(y_2)$, we have $s \in Y_1$. Now, for all $(a_1, \ldots, a_r) \in M^r$, if $a_i = s(x_i)$, for $i \leq r$, then there is some $s' \in Y_1$ such that $a_i = s'(v_i)$ for all $i \leq r$ (this follows from the assumption that (11) holds). But we know that $(s'(v_1), \ldots, s'(v_r)) \in \mathrm{Rel}(X)$, which is what we wanted.

– To see $\mathcal{M}, \mathrm{Rel}(X), (f_j^i)_{i,j}, s \models \neg R\overline{x} \vee f_1\overline{x} = f_2\overline{x}$, assume $\mathcal{M}, \mathrm{Rel}(X), s \models R\overline{x}$. Then note that from $\mathcal{M}, Y \models \theta^*$ we get a split $Y = Y_1 \cup Y_2$ such that $\mathcal{M}, Y_1 \models \bigotimes_{i\leq r} \neg(v_i = x_i)$ and $\mathcal{M}, Y_2 \models y_1 = y_2$. Consider $s' := s(v_i \mapsto s(x_i))_{i\leq r}$. Then $s' \in Y$ because $s \in Y$ and $(s(x_1), \ldots, s(x_r)) \in \mathrm{Rel}(X)$. Because $s'(v_i) = s(x_i)$ for all $i \leq r$, we have $s' \in Y_2$, whence $s'(y_1) = s'(y_2)$, i.e., $\mathcal{M}, (f_j^i)_{i,j}, s \models f_1\overline{x} = f_2\overline{x}$, as we wanted.

– It is left to show $\mathcal{M}, \mathrm{Rel}(X), (f_j^i)_{i,j}, s \models \bigvee_{i\leq q} \alpha_i$, where no α_i mentions R. From $\mathcal{M}, Y \models \theta^*$ we get a split $Y = \bigcup_{i\leq q} Y_i$ such that $\mathcal{M}, Y_i \models \alpha_i^*$ for each $i \leq q$. Because $s \in Y_i$ for some i, we have $\mathcal{M}, s \models \alpha_i$. □

Ludics and Its Applications to Natural Language Semantics

Alain Lecomte[1] and Myriam Quatrini[2]

[1] UMR "Structures Formelles de la Langue", CNRS-Université Paris 8 - Vincennes-Saint-Denis
[2] UMR "Institut de Mathématiques de Luminy", CNRS-Aix-Marseille Université

Abstract. Proofs in Ludics, have an interpretation provided by their *counter-proofs*, that is the objects they interact with. We shall follow the same idea by proposing that sentence meanings are given by the *counter-meanings* they are opposed to in a dialectical interaction. In this aim, we shall develop many concepts of Ludics like *designs* (which generalize proofs), *cut-nets*, *orthogonality* and *behaviours* (that is sets of designs which are equal to their bi-orthogonal). Behaviours give statements their interactive meaning. Such a conception may be viewed at the intersection between *proof-theoretic* and *game-theoretical* accounts of semantics, but it enlarges them by allowing to deal with possibly infinite processes instead of getting stuck to an atomic level when decomposing a formula.

1 Meanings, Proofs and Games

The dominant trend in Natural Language Semantics is based on Frege's conceptions on Logics and Language according to which the meaning of a sentence may be expressed in terms of its truth conditions. There is however an alternative conception according to which we don't find meanings in truth conditions but in *proofs*, particularly expressed by the *Brouwer-Heyting-Kolmogorov* (BHK) - interpretation. This conception has been used in philosophy, linguistics and mathematics. In Natural Language Semantics, it has been for instance developed by Martin-Löf, Sundholm and Ranta ([Martin-Löf 84, Sundholm 86, Ranta 94]), but this framework is limited because proofs are finite objects. At a certain stage of the proof of a formula, atomic formulae are obtained, but *what is the proof of an atomic formula*? Actually, we expect a proof of a sentence to be an object of the same symbolic nature as the sentence. There is no way to escape from language or from mind to directly reach the external world.

Other similar attempts to provide a foundation for meaning in natural language are based on works by Hintikka, Kulas and Sandu ([Hintikka-Kulas 83, Hintikka-Sandu 97]). In their interpretation, meanings are provided by strategies in a *language game*. Their views meet Wittgenstein's according to which *meaning is use* and the use of language is showed in language games. But still those accounts meet difficulties when dealing with atomic sentences: in this case, the logician is obliged to refer to some model (in the traditional sense of Model Theory) in order to evaluate the truth value of an atomic sentence.

Still more seriously, they only take in consideration games which are of a very particular kind: they are oriented towards the notions of *winner*, *winning strategy*, *score*

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 242–255, 2009.

and *pay-off* function, contrarily to what Wittgenstein suggested in his *Philosophische Untersuchungen* when he spoke of games for a very large family (even mere pastimes). Neither Wittgenstein's games do refer to *a priori* rules which would be attached, like in GTS, with logical particles (cf. [Pietarinen 07]).

Finally, none of these alternatives to the truth-conditions based framework ever envisaged to take proofs or games as infinite devices (or partial and underspecified ones) and of course none of these traditions took into account the fact that proofs and strategies can be the *same* objects, simply viewed from different angles.

If this concerns formalized theories of meaning, what to say of theories of meaning which have not been formalized, like that of *argumentative* meaning, in O. Ducrot's sense ([Ducrot 1984]). Ducrot pointed out the so-called *polyphonic* aspect of language, that is the fact that utterrances are not simple statements which are confronted with "reality", but dynamical processes which are oriented towards possible or impossible continuations (for instance *I have a few books* cannot be pursued by **and even none*, while *He read few books* may be). In the same way, *dialogues* may be studied according to what utterrance may be an appropriate reply to another one, and what may not be.

In this paper, we shall present some applications of Ludics to these topics. In a nutshell, proofs in Ludics, have an interpretation provided by their counter-proofs, that is the objects they interact with. We shall follow the same idea by proposing that sentence meanings are given by the *counter-meanings* they are opposed to in a dialectical interaction.

2 Dialogues and Ludics

2.1 Ludics: A Theory of Interaction

Ludics can be sum up as an interaction theory, formulated by J.-Y. Girard ([Girard 01]) as the issue of several changes of paradigms in Proof Theory : from provability to computation, then from computation to interaction. The first change of paradigm arose with the intuitionnistic logic, while the second was due to the development of linear logic.

Starting from a geometrical viewpoint on proofs, which provided an internal approach to the dynamics of proofs, Ludics takes the notion of interaction (that is the cut rule and its process of elimination) as *primitive*. Therefore, it simply starts from *loci*, or adresses (*where* interaction can take place) and *formulae* are given up, at least for a while, since the challenge is to regain them at the output of the construction. *Proto-formulae* are used as mere scaffoldings for building the main objects we shall deal with (the designs).

The central object of Ludics: the design. Using the metaphor of Games, a design can be understood as a *strategy*, i.e. as a set of *plays* (or *chronicles*) ending by answers of Proponent against the moves planned by Opponent. The plays are alternated sequences of *moves* (*actions*). A move is defined as a 3-uple consisting in

- a polarity (positive for Proponent, negative for Opponent),
- an adress or *locus*, coded by a finite sequence of integers (denoted by ξ, ρ, σ ...), where the move is said to be *anchored*,

– a finite set of integers, or *ramification* which indicates the positions which can be reached in one step. A unusual positive move is also possible : the *daïmon*, which may end up a play.

Positions are organized in *forks*, which are presented under the general form: $\Gamma \vdash \Delta$; where Γ and Δ are finite sets of loci such that Γ is either the empty set or a singleton. The fork corresponding to the starting position is called the *base* of the design. When Γ is not empty, the following (opponent) move starts from the only element it contains, and the fork is said to be *negative*, in the other case, Proponent chooses the locus in Δ from where it starts and the fork is said to be *positive*.

Perhaps this may seem not new with regards to *GTS*, let us notice however that moves are defined abstractly, independently from any particular connective or quantifier, and that at each step, the whole history of the previous moves is available.

Now, more importantly, a design may be also seen as a *proof search* in some linear formal system, according to the following methodological choices:

– the object we are building not only provides a proof, but at the same time, contributes to the determination of the formula which is proved. Loci point out the position where such a formula could be located, and at the same time, the "logical" decomposition this formula could have,
– by means of the focalisation property discovered by Andréoli [Andréoli 92] according to which in linear logic, it is always possible to draw a proof by following a strict discipline (focusing) which amounts to grouping together successive blocks of rule applications of the same polarity, it is possible to have only two rules (one positive and one negative).
– it may happen that the research be not successful. In this case, one may give up the proof search, thus using a specific non logical rule (or *paralogism*) : the *daïmon* rule.

A design can therefore be represented by a tree of forks, built by means of three rules :
Daïmon

$$\frac{}{\vdash \Lambda}\,\dagger$$

Negative rule

$$\frac{\ldots \quad \vdash \xi \star J, \Lambda_J \quad \ldots}{\xi \vdash \Lambda}\,(-, \xi, \mathcal{N})$$

Positive rule

$$\frac{\ldots \quad \xi \star i \vdash \Lambda_i \quad \ldots}{\vdash \xi, \Lambda}\,(+, \xi, I)$$

where I and J are finite subsets of $\mathbb{N}$, $i \in I$, with the Λ_i pairwise disjoints, $\mathcal{N}$ is a set (*possibly infinite*) of finite subsets of $\mathbb{N}$, each J of the negative rule being an element of this set, all the Λ_J are included in Λ, and moreover each base sequent is well formed in the sense that all addresses are pairwise disjoint.

The Fax. Since we have not yet introduced formulae, there is no opportunity to use *axiom*-links. Instead, we will have a particular design based on a fork $\xi \vdash \xi'$. Roughly

speaking, this design ensures that both loci ξ and ξ' could be locations of a same formula. That means that as soon as a logical decomposition may be handled on the right hand side, the same may also be handled on the left hand side. Such a design, called $\mathcal{F}ax$, is recursively defined as follows:

$$Fax_{\xi,\xi'} = \dfrac{\dfrac{}{Fax_{\xi'_i,\xi_i}}}{\dfrac{\dots \quad \xi' \star i \vdash \xi \star i \quad \dots}{\vdash \xi \star J, \xi'}\,(+,\xi',J)}\,\dots}{\xi \vdash \xi'}\,(-,\xi,\mathcal{P}_f(\mathbb{N}))$$

At the first (negative) step, the negative *locus* is distributed over all the finite subsets of $\mathbb{N}$, then for each set of addresses (relative to some J), the positive locus ξ' is chosen and gives rise to a subaddress $\xi' \star i$ for each $i \in J_k$, and the same machinery is relaunched for the new loci obtained.

Defining interaction. Interaction is concretely expressed by a coincidence of two loci in dual position in the bases of two designs. This creates a dynamics of rewriting of the cut-net of the two designs, called, as usual, *normalisation*. We sum up this process as follows: the cut link is duplicated and propagates over all immediate *subloci* of the initial cut *locus* as long as the action anchored on the positive fork containing the cut-locus corresponds to one of the actions anchored on the negative one. The process terminates either when the positive action anchored on the positive cut-fork is the *daïmon*, in which case we obtain a design with the same base as the starting cut-net, or when it happens that in fact, no negative action corresponds to the positive one. In the later case, the process fails (or *diverges*). The process may not terminate since designs are not necessarily finite objects.

When the normalization between two designs $\mathcal{D}$ and $\mathcal{E}$ (respectively based on $\vdash \xi$ and $\xi \vdash$) succeeds, the designs are said to be *orthogonal*, and we note: $\mathcal{D} \perp \mathcal{E}$. In this case, normalization ends up on the particular design :

$$\dfrac{}{\vdash}\,[\dagger]$$

Let $\mathcal{D}$ be a design, $\mathcal{D}^\perp$ denotes the set of all its orthogonal designs. It is then possible to compare two designs according to their counter-designs. We set $\mathcal{D} \prec \mathcal{E}$ when $\mathcal{D}^\perp \subset \mathcal{E}^\perp$.

The separation theorem [Girard 01] ensures that this relation of preorder is an order, so that a design is exactly defined by its orthogonal.

Behaviours. One of the main virtues of this "deconstruction" is to help us rebuilding Logic.

- Formulas are now some sets of designs. They are exactly those which are closed (or stable) by interaction, that is those which are equal to their *bi-orthogonal*. Technically, they are called *behaviours*.
- The usual connectives of Linear Logic are then recoverable, with the very nice property of *internal completeness*. That is : the bi-closure is useless for all linear connectives. For example, every design in a behaviour $\mathbf{C} \oplus \mathbf{D}$ may be obtained by taking either a design in $\mathbf{C}$ or a design in $\mathbf{D}$.

– Finally, *proofs* will be now designs satisfiying some properties, in particular that of not using the daïmon rule.

2.2 Ludics as a Formal Framework for Dialogues

Concerning dialogues, let us focalize on the mere *supports* of the interaction. That is the *locus* where a speech turn is anchored (among the *loci* previously created) and the *loci* that it creates, which are also those which may be used later on.
Because Ludics may display the history of the dialogue by means of *chronicles*, and it takes into account the strategies of any speaker by means of *designs*, it allows us to see a dialogue as the result of an interaction between the strategies of two speakers. In that case, the rules have the following interpretation:

– when being *active* (that is using a positive rule), a speaker chooses a *locus* and therefore has an active role,
– when being *negative* (that is using a negative rule), s/he has no choice and has a passive role

If, therefore, positive steps are understood as moves where the intervener asks a question or makes an assertion, and negative steps as moves where s/he is apparently passive, recording an assertion and planning a further reply, positive actions of one speaker are not opposed to positive actions of the other one (as it is the case in most formal accounts of dialogue, even the logical ones) but to negative ones of the other. This point meets an important requirement formulated by Ducrot according to whom "the semantic value of an utterrance is built by allusion to the possibility of another utterrance (the utterrance of the Other speaker)".

Examples

– The following example is deliberately simple, and only given for a pedagogic purpose.
Let us consider the following dialogue between Annie and Barbara:
A : did you meet some friends yesterday evening to the party ? **B** : I only saw Bruno and Pierre. **A** : Was Pierre still as nice as during the last year ? **B** : Yes, he did. **A** : That is what I wanted to know.
Such an exchange is represented by an interaction between two designs : one is seen from the point of view of **A** and the other from the point of view of **B**:

From A's point of view	From B's point of view
$\vdash 0.1.1.1.1$	$0.1.1.1.1 \vdash$
$0.1.1.1 \vdash$	$\vdash 0.1.1.1$
$\vdash 0.1.1, 0.1.2$	$0.1.1 \vdash \quad 0.1.2 \vdash$
$0.1 \vdash$	$\vdash 0.1$
$\vdash 0$	$0 \vdash$

The trace of the interaction (the cut between the two loci 0) is the alternated sequence of actions: $(+, 0, \{1\})(-, 0.1, \{1, 2\})(+, 0.1.1, \{1\})(-, 0.1.1.1, \{1\})\dagger$.
In this case the normalisation ends up on the daïmon. The interaction converges.

– The second example is taken from Schopenhauer's "Dialectica eristica" (or "The Art of Always Being Right") which provides a series of so-called *stratagems* in order to be always right in a debate. It formalizes the first given stratagem.
"I asserted that the English were excellent in drama. My opponent attempted to give an instance of the contrary, and replied that it was a well-known fact that in opera, they were bad. I repelled the attack by reminding him that, for me, dramatic art only covered tragedy and comedy ...."
We give an account of this dialogue by the following interaction:

$$\dfrac{\dfrac{\dfrac{\vdash \xi.1.1 \qquad \vdash \xi.1.2}{\xi.1 \vdash}\,C}{\vdash \xi}\,A \qquad \dfrac{\dfrac{\xi.1.3 \vdash}{\vdash \xi.1}\,B}{\xi \vdash}}{}$$

Where the action A corresponds with the claim: "The English are excellent in drama" ; the action B with "I disagree, it is a well-known fact that in opera, they could do nothing at all." and the action C with "But by dramatic art, I only mean tragedy and comedy."
Of course, the net built with these two designs does not converge. In fact, things don't happen this way: initially, the set of loci the first speaker has in mind could also cover *opera*. What happens when willing to repel the attack is retracting one branch (or replay the game according to a different strategy). This leads us to enter more deeply into the decomposition of dialogues and in what we consider as *units* of action.

While, at the most elementary level, which is relevant as long as the dialogues we consider are simple (for instance exchanges of information), the interaction is between elementary actions, those elementary actions are replaced by (sub)-designs as soon as we are concerned by dialogues of a more complex nature like controversies.

– A third example comes from Aristotle's *Sophistical Refutations*, where it is given the name *multiple questions*.
Let us imagine a judge asking a man the question:
"Do you still beat your father ?".
The judge asks a question that presupposes something that has not necessarily been accepted by the man. S/he imposes to him the following implicit exchange:
- "Do you beat your father?" - " Yes" - " Do you stop beating him ?".
This exchange between the judge J and the man D must be represented by the following interaction :

$$\dfrac{\dfrac{\dfrac{\xi.\mathbf{0}.\mathbf{1}.\mathbf{0} \vdash}{\vdash \xi.\mathbf{0}.\mathbf{1}} \qquad \vdash \xi.0.2}{\xi.\mathbf{0} \vdash}}{\vdash \xi}\,J \qquad \dfrac{\dfrac{\vdash \xi.0.1.0}{\xi.0.1 \vdash}}{\vdash \xi.0}}{\xi \vdash}\,D$$

In fact, the judge utterance: - *"Do you still beat your father ?"* contains what we call nowadays a *presupposition*. It can't therefore be represented by *a single action*, but by *the whole chronicle*: $(+, \xi, \{0\})\ (-, \xi.0, \{1\})\ (+, \xi.0.1, \{0\})$. This enables us to give an account of the fact that one of the *loci* where the interaction might continue is in fact not available ; in some sense the action giving this possibility is skipped, some successive one is immediately proposed and, by this way, constrains the answers.

The ludical approach thus allows us to get a formalized conception of stratagems and fallacies, something which appeared out of reach for many researchers (see for instance [Hamblin 70]). Moreover, we claim that it could improve some issues in formal semantics, like we try to show it in the following section.

3 Logical Forms and Ludics

In the sequel, we propose a conception of interactive meaning based on Ludics. In the same way a design is defined by its orthogonal (according to the separation theorem), we may postulate that the meaning of a sentence is given by the set of all its dual sentences: that is all the sentences with which the interaction converges. For this purpose, we associate a behaviour or a family of behaviours with a sentence. Such behaviours are built in a compositional way, like in standard formal semantics, but their ultimate components are neither *atoms* nor *atomic formulae*, like in the Intensional Logic Montague was using. Let us underline the points which are slightly different and new and which could favourably extend the standard models of semantics:

- The fact that the mathematical object associated with the meaning of a sentence may be *more and more refined* seems to us very important. Such an objective is realized because of the order on designs involved by the *separation theorem*, which enables one to explore more and more precisely the argumentative potential of a sentence. Moreover, new designs may always be added to such an object, thus enlarging our conception of meaning.
- The fact that Ludics strictly encompasses logic and that logical concepts like formulas, proofs or connectives are defined in a world which is larger than the strictly logical one (let us remember that we have *paralogisms* like the *daïmon*, and counter-proofs in that world!) makes us to expect more freedom in defining "logical" forms. For instance it may be the case that behaviours are composed by means of a non-logical operator (but which could nevertheless be interpreted).

The following example illustrates a classical problem of ambiguty (*scope* ambiguity).

3.1 Meaning through Dual Sentences

The meaning of a sentence is given by all the utterances which correctly interact (that means : *converge*) with it.

Let us consider the statement (from now on denoted by S): "Every linguist speaks some african language". Usually two logical forms can be associated with such a sentence S, depending on whether *some* has the narrow or the wide scope. Namely:

$$S_1 = \quad \forall x(L(x) \Rightarrow \exists y(A(y) \wedge P(x,y)))$$
$$S_2 = \quad \exists y(A(y) \wedge \forall x(L(x) \Rightarrow P(x,y)))$$

where $L(x)$ means "x is a linguist" , $A(y)$ means "y is an african language" and $P(x,y)$ means "x speaks y".
When "some" has the narrow scope, we assume that the logical form converges with the LF of sentences like:

(1) There is a linguist who does not know any african language.
(2) Does even John, who is a linguist, speak an african language ?
(3) Which is the African language spoken by John ?

On the opposite, if "some" has the wide scope, the logical form converges with :

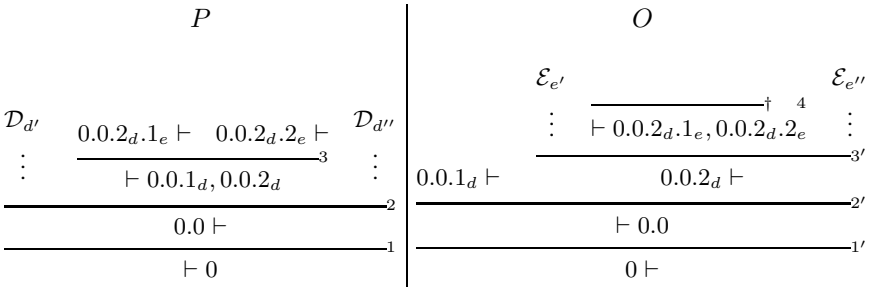
(4) There is no african language which is spoken by all the linguists.
(5) Which african language every linguist speaks ?

3.2 Meaning as a Set of Justifications

We materialize the claim according to which meaning is equated with a set of dual sentences by associating with the meaning of S a set of designs. Such designs may be seen as *justifications* of S. That is the supports of the dialogues during which a speaker P asserts and justifies the statement S against an adressee O who has several tests at his/her disposal.

Let us make such a design, based on the arbritary fork $\vdash 0$, more precise:

- the first action corresponds to the assertion of S. Its *ramification* is a singleton ; only one locus is created for continuing the interaction. Nevertheless, the speaker who has to anticipate the reactions of his/her adressee is committed to one of the readings of S. S/he is ready to assume one of the two possibilities, the wide or the narrow scope for "some". This is taken into account by distinguishing between two possible first actions, that we symbolize for instance by $(+, 0, \{0\})$ and $(+, 0, \{1\})$. It is then possible to distinguish between two kinds of designs, considered as justifications of S according to the choice of the first action.
- let us for instance focus on the first reading of S. We then simulate an interaction between P and O who tries to negate P's claim:



The normalisation stages may be commented as follows:

1 P asserts S and is ready to continue the interaction with the first reading of S
1' O records the claim made by P and is ready to answer it. Notice that if O had been ready to answer according to the second reading, its action would have been $(-, 0, \{1\})$ and the interaction would have diverged
2 P is ready to give justifications for any individual : d,d',...
2' O proposes an individual d (arguing that d is a linguist (localized in $0.0.1_d$) and that d doesn't know any african language (localized in $0.0.2_d$))
3 P exhibits some language e (arguing that e is an african language and d speaks e)
3' at the same time, O is ready to receive such a claim by P for some language among e',e, e'' ...
4 if P has given some language e such that d speaks it, O may be ready to give up.

Thus, the interaction between "Every linguist speaks some african language" and the attempt to negate it "There is some linguist which doesn't speak any african language" normalizes.

Let us denote by $\mathcal{D}$ the foregoing design of P. We could also find another design as justification of S with its first reading : P may ask to **check** if d is really a linguist, O may ask to **check** if d really speaks e and so on thus providing a deeper interaction. Further exchanges may enter into debates on what it means for a person to be a linguist, or on what it means for a language to be an african one, or on what it means for a person and a language to be such that the person speaks the language and so on...

In any way, if $\mathbb{S}_1$ denotes the set of designs representing the first reading of S and if $\mathbb{S}_2$ denotes the set of designs representing the second one, the set of designs representing the meaning of S is the union of both sets : $\mathbb{S} = \mathbb{S}_1 \cup \mathbb{S}_2$.

3.3 Meaning as Behaviour

The previous attempt to associate a set of design with the meaning of S is still general and imprecise. $\mathcal{D}$ actually belongs to the following behaviour[1]:

$$\forall x(\downarrow \mathbb{L}(x) \multimap \exists y(\downarrow \mathbb{A}(y) \otimes \downarrow \mathbb{P}(x, y)))$$

provided that $\mathbb{L}(x)$, $\mathbb{A}(y)$ and $\mathbb{P}(x, y)))$ are behaviours. Indeed, following the correspondance between designs and proofs of the hypersequentialized polarised linear logic H which is given in the annex, the design $\mathcal{D}$ may be seen as an attempt to prove the formula $S = S_1 \oplus S_2$ where S_1 and S_2 are the (proto) - formulas associated with the first and second reading of S in their linear and hypersequentialized formulations :

$$\dfrac{\dfrac{\mathcal{D}_{d'} \quad \dfrac{\downarrow A^{\perp}(e_d) \vdash \qquad \downarrow P^{\perp}(d, e_d) \vdash}{\vdash \downarrow L^{\perp}(d), \exists y(\uparrow A(y) \otimes \uparrow P(d, y))} \quad \mathcal{D}_{d''}}{(\forall x(\uparrow L(x) \multimap \exists y(\uparrow A(y) \otimes \uparrow P(x, y))))^{\perp} \vdash}}{\vdash S}$$

[1] $\forall$ and $\exists$ are used here because of their intuitive appeal, but in fact they stand for the generalized additives connectives $\&_x$ and $\oplus_y$ (cf. annex). There is nevertheless a slight difference between both pairs of concepts: strictly speaking, in Ludics the correct use of first order quantifiers with regards to mathematical formulas would involve a uniformity property ([Fleury-Quatrini 04]) which is neither relevant nor satised here.

We retrieve the semantical notion of "logical form" but resting on *behaviours* instead of, simply, logical formulae. There are finally two possible ways to associate a behaviour with a sentence:

- either, we can consider that the design obtained (as in the previous section) as a minimal justification of S may generate a behaviour associated with S. Thus $\mathbb{S}_{gen} = \mathcal{D}^{\perp\perp}$.
- or we can consider that the behaviour associated with S corresponds to the linear formula (in an hypersequentialised formulation):

$$\mathbb{S} = (\forall x(\downarrow \mathbb{L}(x) \multimap \exists y(\downarrow \mathbb{A}(y) \otimes \downarrow \mathbb{P}(x,y)))) \oplus \exists y(\downarrow \mathbb{A}(y) \otimes \forall x \uparrow (\downarrow \mathbb{L}(x) \multimap \downarrow \mathbb{P}(x,y))).$$

The later interpretation of S's meaning is in fact a *family of behaviours* because $\mathbb{S}$ depends on the behaviours $\mathbb{L}(x)$, $\mathbb{A}(y)$ and $\mathbb{P}(x,y)$)).

Remark 1. *As a* logical formula*, S is seen as the disjunction of S_1 and S_2 , namely as the formula $S = S_1 \oplus \downarrow S_2$, and as a* behaviour*, seen as the union*[2] *of the two behaviours associated with the two terms of the disjunct. Hence we get a logical account of the fact that interaction may activate only one of both logical sub-formulas, depending on the scope of "some".*

Remark 2. *The behaviour $\mathbb{S}_{gen}$ contains all the behaviours logically built from the behaviours associated with the elemantary pieces $L(x)$, $A(y)$ and $P(x,y)$. This way we get a first (and still rough) account of the logical particles of meaning.*

Finally, Ludics enables us to go further into the specification of the logical form.

Decomposing "atomic formulas"

1. It is of course possible to consider the leaves of a decomposition as atomic formulae, if decomposition ends up. In this case, they are seen as *data items*[3]
   We can thus consider the following design $\mathcal{D}'$ as a justification of S :

$$\dfrac{\mathcal{D}_{d'} \quad \dfrac{\dfrac{\dfrac{}{\vdash A(e_d)}\emptyset}{\downarrow A^{\perp}(e_d) \vdash} \quad \dfrac{\dfrac{}{\vdash P(d,e_d)}\emptyset}{\downarrow P^{\perp}(d,e_d) \vdash}}{\vdash \downarrow L^{\perp}(d), \exists y(\uparrow A(y) \otimes \uparrow P(d,y))} \quad \mathcal{D}_{d''}}{(\forall x(\uparrow L(x) \multimap \exists y(\uparrow A(y) \otimes \uparrow P(x,y))))^{\perp} \vdash}$$

   Let us remark that $\mathcal{D}'$ is more defined than $\mathcal{D}$. In Ludics this means that $\mathcal{D}^{\perp} \subset \mathcal{D}'^{\perp}$ and this may be understood here that the justification is more informative, more precise.

[2] This is one of the mains results of Ludics: the internal completeness ensures that the elementary operation of union is enough to obtain all the designs of the disjunction.

[3] in Ludics this is possible by means of the use of the linear multiplicative constant 1.

2. But we may also consider that $L(x)$, $A(y)$ and $P(x, y)$ are *still decomposable*. That amounts to recognize that $\mathbb{S}_1$ contains other designs: all those which are more defined than $\mathcal{D}$. Designs *more defined than* $\mathcal{D}$ are built on the same schema than $\mathcal{D}'$ but instead of ending on the the empty set, they continue on non empty ramifications, thus allowing the exploration of $A(e_f)$ or $P(f, e_f)$ which were alleged atomic formulae in the previous designs.

The vericonditional interpretation is here retrieved as an indirect (and secondary) consequence of our "(para)proofs as meanings"[4] interpretation because now, $\mathcal{D}'$ is really a proof provided that $A(f)$ and $P(f, e_f)$ are either data items, that is the true linear formula **1** or are provable when they are decomposable.

3.4 How to Go Further ?

Towards speech acts - and the use of $\mathcal{F}$ax. Instead of simple yes/no questions, where convergence occurs for "yes" and divergence for "no", we may take so called *wh*-questions into consideration, for example "which is the african language that John speaks ?". In this case *we expect that the interaction has the answer as its by-product (or its side effect)*.

To reach this goal, let us associate with such a question (that we may see as a *speech act*) a design in which *there is a locus for storing the answer*. In our formulation of designs as HS-paraproofs, this question will be associated with a paraproof of the sequent $S \vdash A$ where A is a formula equal to $\uparrow A_1 \oplus \cdots \oplus \uparrow A_n$ corresponding to the logical form of " is some african language" (afar, peul, ewe, ewondo...). A complex design using $\mathcal{F}ax$ will be associated with the question "which is the african language that John speaks ?" . The result of the interaction of this design with $\mathcal{D}'$ is : $\dfrac{\dfrac{\overline{\vdash A_e}^{\emptyset}}{\downarrow A_e^{\perp} \vdash}}{\vdash A}$ which can be read as " A_e is this african language" (where A_e is the african language that John speaks (in $\mathcal{D}'$)).

In our opinion, this suggests a way to perform in Ludics a unified treatment of Logical Forms and Speech Acts. At the same time, this underlines the richness of the ludical framework to give an account of the interactions in language.

4 Conclusion

In this paper, we tried to give an account of Ludics and of the new way it allows to specify Meaning in Language: not by considerations on truth conditions but by using the important concept of *interaction*. To access the meaning of a sentence is mainly to know how to question, to answer to or to refute this sentence, and to know how to extend the discourse (or the dialogue) to which it belongs. In such a conception, the meaning of a sentence is a moment inside an entire process which coud be conceived as infinite (if

[4] In the opposition of two processes of proof search, both cannot be "real" proofs, it is the reason why we call them paraproofs.

for instance we admit that the interpretation or the argumentation process with regards to any statement is potentially infinite). Ludics gives a precise form to these views by means of the notions of *normalization* and *behaviour*.

Otherwise, the emphasis put on *loci* has, as a valuable consequence, the fact that we may conceive several instances of the same *sign* (a sentence, a word etc.) as having various meanings, according to the location it has in a discourse or a dialogue, thus giving suggestions for dealing with many rhetorical figures (and *fallacies*). The infinite design $\mathcal{F}ax$ allows to delocate such meanings but its use is not mandatory. Moreover, the fact (not much developed in this extended abstract) that a design may be viewed either as a kind of proof (in a syntactic setting of the framework) or as a game (in a semantic setting of it) provides us with interesting insights on Pragmatics and Wittgensteinian language games. In a pragmatic theory of presupposition, for instance, *presupposing* implies making an assertion where the hearer has no access to a previous step made by the speaker, if (s)he rejects this step, (s)he makes the process to diverge. Other "games" may be explored. Wittgenstein for instance quoted *elicitation*, that is the way in which somebody may obtain an answer to a question. Every time, $\mathcal{F}ax$ is used to transfer a meaning from a location to another one (for instance from the discourse or the brain of the other speaker to the one of the eliciter). Those games may be envisaged without any kind of "winning strategy". In a speech act seen as a game, there is no win, simply the appropriate use of some designs in order to reach an objective (which may be a common one). Future works will be done in those directions.

References

[Andréoli 92] Andréoli, J.-M.: Logic Programming with Focusing Proofs in Linear Logic. The Journal of Logic and Computation 2, 3, 297–347 (1992)

[Curien 2004] Curien, P.-L.: Introduction to linear logic and ludics, part I and II, to appear, http://www.pps.jussieu.fr/~curien/LL-ludintroI.pdf

[Ducrot 1984] Ducrot, O.: Le dire et le dit, Editions de Minuit, Paris (1984)

[Fleury-Quatrini 04] Fleury, M.-R., Quatrini, M.: First order in Ludics. Mathematical Structures in Computer Science 14(2), 189–213

[Girard 99] Girard, J.-Y.: On the Meaning of Logical Rules-I in Computational Logic. In: Berger, U., Schwichtenberg, H. (eds.). Springer, Heidelberg (1999)

[Girard 01] Girard, J.-Y.: Locus Solum Mathematical Structures in Computer. Science 11, 301–506 (2001)

[Girard 03] Girard, J.-Y.: From Foundations to Ludics. Bulletin of Symbolic Logic 09, 131–168 (2003)

[Girard 06] Girard, J.-Y.: Le Point Aveugle, vol. I, II. Hermann, Paris (2006)

[Hamblin 70] Hamblin Fallacies, C.-L.: Vale Press, Newport News (republished, 2004)

[Hintikka-Kulas 83] Hintikka, J., Kulas, J.: The Game of Language: Studies in Game Theoretical Semantics and its Applications. D. Reidel (1983)

[Hintikka-Sandu 97] Hintikka, J., Sandu, G.: Game Theoretical Semantics. In: Van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language, ch. 6, Elsevier, Amsterdam (1997)

[Martin-Löf 84] Martin-Löf, P.: Intuitionistic Type Theory, Bibliopolis, Naples (1984)

[Pietarinen 07] Pietarinen, A.-V.: Game Theory and Linguistic Meaning. Elsevier, Amsterdam (2007)
[Ranta 94] Ranta, A.: Type-Theoretical Grammar. Oxford University Press, Oxford (1994)
[Schopenhauer] Schopenhauer, A.: The Art of Always Being Right
[Sundholm 86] Sundholm, G.: Proof Theory and Meaning. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. III, pp. 471–506. D. Reidel, Dordrechtz (1986)
[Tronçon 06] Tronçon, S.: Dynamique des démonstrations et théorie de l'interaction, PhD thesis, Université d'Aix-Marseille (2006)
[Wittgenstein 53] Wittgenstein, L.: Philosophische Untersuchungen. Blackwell, Malden (1953)

A A Hypersequentialized Version of the Linear Sequent Calculus

We give here a short presentation of a hypersequentialized version of linear calculus, which enables one to manipule the designs as (para)proofs of a logical calculus.

A.1 Formulas and Sequents

By means of polarity, we may simplify the calculus by keeping *only positive formulae*. Of course, there are still negative formulae... but they are simply put on the left-hand side after they have been changed into their negation. Moreover, in order to make paraproofs to look like sequences of alternate steps (like it is the case in ordinary games), we will make blocks of positive and of negative formulae in such a way that each one is introduced in only one step, thus necessarily using *synthetic connectives*. Such connectives are still denoted $\oplus$ and $\otimes$ but are of various arities. We will distinguish the case where both $\oplus$ and $\otimes$ are of arity 1 and denote it $\downarrow$.

- The only linear formulae which are considered in such a sequent calculus are built from the set P of linear constants and propositionnal variables according to the following schema :

$$F = P | (F^{\perp} \otimes \cdots \otimes F^{\perp}) \oplus \cdots \oplus (F^{\perp} \otimes \cdots \otimes F^{\perp}) | \downarrow F^{\perp}$$

– The sequents are **denoted** $\Gamma \vdash \Delta$ where Δ is a multiset of formulas and Γ contains at most a formula.

A.2 Rules

– There are some axioms (logical and non logical axioms):

$$\frac{}{P \vdash P} \qquad \frac{}{\vdash 1} \qquad \frac{}{\vdash \downarrow T, \Delta} \qquad \frac{}{\vdash \Delta}\,\dagger$$

where P is a propositionnal variable ; 1 and T are the usual linear constants (respectively positive and negative).

– The "logical" rules are the following ones :
Negative rule

$$\frac{\vdash A_{11}, \ldots, A_{1n_1}, \Gamma \quad \ldots \quad \vdash A_{p1}, \ldots, A_{pn_p}, \Gamma}{(A_{11} \otimes \cdots \otimes A_{1n_1}) \oplus \cdots \oplus (A_{p1} \otimes \cdots \otimes A_{pn_p}) \vdash \Gamma}$$

Positive rule

$$\frac{A_{i1} \vdash \Gamma_1 \quad \ldots A_{in_i} \vdash \Gamma_p}{\vdash (A_{11} \otimes \cdots \otimes A_{1n_1}) \oplus \cdots \oplus (A_{p1} \otimes \cdots \otimes A_{pn_p}), \Gamma}$$

where $\cup \Gamma_k \subset \Gamma$ and for $k, l \in \{1, \ldots p\}$ the $\Gamma_k \cap \Gamma_l = \emptyset$.

A.3 Remarks on Shifts

Using the shift is a way to break a block of a given polarity. Separate steps may be enforced by using the *shift* operators $\downarrow$ and $\uparrow$ which change the negative (resp. positive) polarity into the positive (resp. negative) one. The rules introducing such shifted formulas are particular cases of the positive and the negative one:

$$\frac{A^{\perp} \vdash \Gamma}{\vdash \downarrow A, \Gamma}\,[+] \qquad\qquad \frac{\vdash A^{\perp}, \Gamma}{\downarrow A \vdash \Gamma}\,[-]$$

where A is a negative formula.

Example. In a block like $A \otimes B \otimes C$ in principle, A, B and C are negative, but if we don't want to deal with A, B, C simultaneously, we may change the polarity of $B \otimes C$ (which is positive) and make it negative by means of $\uparrow$. We write then $A \otimes \uparrow (B \otimes C)$. Compare the two following partial proofs, where (1) does not use any shifts and (2) uses one :

instead of (1): $\dfrac{A^{\perp} \vdash \quad B^{\perp} \vdash \quad C^{\perp} \vdash}{\vdash A \otimes B \otimes C}$ we get (2) : $\dfrac{A^{\perp} \vdash \quad \downarrow (B \otimes C)^{\perp} \vdash}{\vdash A \otimes \uparrow (B \otimes C)}$ $\dfrac{B^{\perp} \vdash \quad C^{\perp} \vdash}{\vdash B \otimes C}$

We may use the notation $\oplus_y$ (and dually $\&_x$) instead of $F_{y_1} \oplus \cdots \oplus F_{y_n}$ or simply $\exists y$ (dually $\forall x$) when it is clear in context that y belongs to a finite set.

Spoilt for Choice: Full First-Order Hierarchical Decompositions

Sebastian Link*

School of Information Management
Centre for Logic, Language and Computation
Victoria University of Wellington, New Zealand
sebastian.link@vuw.ac.nz

Abstract. Database design aims to find a database schema that permits the efficient processing of common types of queries and updates on future database instances. Full first-order decompositions constitute a large class of database constraints that can provide assistance to the database designer in identifying a suitable database schema.

We establish a finite axiomatisation of full first-order decompositions that reflects best database design practice: an inference engine derives all potential candidates of a database schema, but the final choice remains with the database designer.

Keywords: Database Decomposition, Database Constraint, Axiomatisation, Propositional Logic.

1 Introduction

Modern database management systems provide commensurate tools to store, manage and process different kinds of data. The core of these systems still relies on the sound technology that is based on the relational model of data [8]. From the perspective of finite model theory, a relational database is a finite structure over a relational signature [24]. Relations permit the storage of inconsistent data, i.e., data that violate conditions which every legal database instance ought to satisfy. Consequently, semantic constraints are specified by the database designer in order to restrict the databases to those which are considered meaningful to the application at hand. During database normalisation join-related constraints are explored to minimise data redundancy for efficient means of updating. In practice, most normalised schemata are subject to denormalisation in order to facilitate the efficient processing of the most common types of queries. The quality of the target database depends on the ability to reason correctly and appropriately about database constraints [14,28].

Full first-order hierarchical decompositions (FOHDs) constitute a large class of relational constraints [9]. A relation is a model of an FOHD when it is the

* This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 256–267, 2009.

Table 1. A relation over WORK and two of its projections

Employee	Child	Insurance	Salary	Year
Al	Bud	AMI	52k	2009
Al	Kelly	AMI	52k	2009
Al	Kelly	State	52k	2009
Al	Bud	State	52k	2009

Employee	Child
Al	Kelly
Al	Bud

Employee	Insurance	Salary	Year
Al	AMI	52k	2009
Al	State	52k	2009

natural join over at least two of its projections that all share the same join attributes. More precisely, a relation r satisfies the FOHD $X : [Y_1 \mid \cdots \mid Y_k]$ when r is the natural join over the projections $r[XY_i]$ of r to the attribute sets XY_i for all $i = 1, \ldots, k$.

Example 1. As a running example we consider the relation schema WORK comprising the attributes *Employee*, *Salary*, *Year*, *Insurance* and *Child*. Intuitively, any row over the table WORK collects information about an employee, the salary of this employee in a certain year, an insurance that the employee has taken out, and a child of the employee. It appears that the information on the child of any employee is independent of the information on the insurance, salary and year of the same employee. This separation of facts can be modelled by the FOHD *Employee* : [*Child* | *Insurance,Salary, Year*]. We may also choose to specify the FOHD *Employee* : [*Salary, Year* | *Child,Insurance*] indicating that the information on the year's salary depends only on the employee independently of the employee's information on children and insurances. The WORK-relation in Table 1 satisfies the first FOHD since it is the natural join over its two projections on {*Employee,Child*} and {*Employee,Insurance,Salary, Year*}. □

Database constraints interact with one another. In fact, a constraint is implicitly specified if it is satisfied by any database that satisfies all the constraints that have been specified explicitly. A fundamental problem is the determination of such implicit knowledge. An axiomatisation for the implication of database constraints can form the basis of an enumeration algorithm that lists all consequences. In practice, such an enumeration is often desirable to validate explicit knowledge. For instance, the FOHD *Employee* : [*Child* | *Insurance* | *Salary, Year*] is implied by the the two FOHDs from Example 1.

FOHDs tell the database designer which attribute sets can form independent information units, e.g. {Employee, Child} and {Employee, Salary, Year, Insurance} based on the FOHD *Employee* : [*Child* | *Insurance,Salary, Year*]. Moreover, the original notion of an FOHD, as introduced by Delobel [9], also tells us in which order the separation of these information units takes place.

Example 2. The FOHD *Employee* : [*Child* | *Insurance* | *Salary, Year*] instructs us to first decompose the relation schema WORK from Example 1 into {*Employee*, *Child*} and {*Employee,Insurance,Salary, Year*}, and subsequently to decompose the latter schema into {*Employee,Insurance*} and {*Employee,Salary, Year*}. □

Intuitively, the decomposition can be processed in an arbitrary order and not just in the order that is indicated by a given FOHD. That is, for any permutation π the FOHD $X : [Y_{\pi(1)} \mid \cdots \mid Y_{\pi(k)}]$ is implied by the FOHD $X : [Y_1 \mid \cdots \mid Y_k]$. Syntactically, this can be represented as the inference rule

$$\frac{X : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_{\pi(1)} \mid \cdots \mid Y_{\pi(k)}]}$$

known as the *permutation rule* [9,29].

Example 3. Given the FOHD *Employee* : [*Child* | *Insurance* | *Salary, Year*] we apply the permutation rule to infer *Employee* : [*Insurance* | *Salary, Year* | *Child*]. This FOHD instructs us to first decompose the relation schema WORK from Example 1 into {*Employee*,*Insurance*} and {*Employee*,*Salary*, *Year*,*Child*}, and subsequently to decompose the latter schema into {*Employee*,*Salary*, *Year*} and {*Employee*,*Child*}. □

The final database schema of a decomposition is invariant under the order in which attribute sets on the right-hand side of an FOHD appear. In practice, however, it is rarely the case that the final database schema of a decomposition is chosen as the layout of the target database. The reason for this is that normalised database schemata only guarantee the efficient processing of updates [30]. Very commonly, the efficient processing of common queries outweighs the maintenance of database constraints and hence, databases are denormalised.

Example 4. Consider the two database schemata $\mathcal{D}_1$:

{*Employee*,*Child*}, {*Employee*,*Insurance*}, {*Employee*,*Salary*, *Year*}

and $\mathcal{D}_2$:{*Employee*,*Child*,*Insurance*}, {*Employee*,*Salary*, *Year*}. The schema $\mathcal{D}_1$ provides a good choice when the most common types of queries do not require any joins between the three relation schemata and/or updates of either Child-, or Insurance- or Salary- and Year-information based on Employee-values are common. In that case no maintenance of FOHDs on any of the relation schemata is necessary. The second schema $\mathcal{D}_2$ may become preferable if common queries require a combination of Child- and Insurance-values, e.g. what is the most common insurance of employees that have at least two children? The efficient processing of these queries may outweigh the maintenance of the FOHD *Employee* : [*Child* | *Insurance*] on {*Employee*,*Child*,*Insurance*}. □

We note that denormalised database schemata occur in intermediate steps of the normalisation process, e.g. {*Employee*,*Salary*, *Year*,*Child*} in Example 3. Such denormalised schemata may only be discovered due to the inference of an implicitly specified FOHD. In particular, the order in which the attribute sets occur on the right-hand side of an FOHD does seem to matter after all.

However, in database practice it is the database designer who chooses the final layout of the database. Consequently, the database designer chooses the order in which the information units are separated from one another. Hence, we would

like to gain complete knowledge about these information units, i.e., the minimal sets of attributes that are independent of one another.

Contributions. In this paper, we will develop a theory that reflects this common practice in database design. First, we develop a theory based on the original notion of an FOHD. We establish an axiomatisation in which it is always possible to delay an application of the permutation rule to the very last step of the inference. Consequently, the decision on the order in which the information units are separated from one another is delayed until the end of the design process (this reflects the choice of the database designer after they become aware of all possibilities for separating the information). If it was not always possible to shift the permutation rule to the very last step of an inference, then one may argue that database designers are limited in their choices for the final layout of the database. That is, the inference engine pre-determines an order in which the information units are to be separated, without any good reason.

Secondly, we introduce order-invariant hierarchical dependencies (OIHDs) which do not take the order of a decomposition into account. Intuitively, an OIHD $X : \{Y_1, \ldots, Y_k\}$ represents the set

$$\{X : [Y_{\pi(1)} \mid \cdots \mid Y_{\pi(k)}] : \pi \text{ is a permutation on } \{1, \ldots, k\}\}$$

of FOHDs. The OIHD *Employee*:{{*Insurance*}, {*Salary, Year*},{*Child*}} represents the 6 different FOHDs where the left-hand side is *Employee* and the right-hand side is a permutation of {*Insurance*}, {*Salary, Year*}, and {*Child*}, for example. We establish an axiomatisation for the implication of OIHDs which contains exactly the inference rules adapted from our axiomatisation of FOHDs, but without the permutation rule. This is a further explanation of our intuition about the design process in practice: an inference engine mechanically determines those information units that can be separated, and, subsequently, the database designer decides how the actual separation will be implemented.

Finally, we show that correspondences between dependencies and fragments of propositional logic [19,27] do not carry over to full-first order decompositions.

Organisation. We repeat concepts of the relational model of data in Section 2, in particular FOHDs, their semantic implication and syntactical inference. In Section 3 we establish an axiomatisation of FOHDs that reflects the role of the permutation role. We study order-invariant hierarchical dependencies in Section 4 and comment on their relationships to propositional logic in Section 5. We conclude in Section 6.

2 Full First-Order Decompositions

Let $\mathfrak{A} = \{A_1, A_2, \ldots\}$ be a (countably) infinite set of symbols, called *attributes*. A *relation schema* is a finite set $R = \{A_1, \ldots, A_n\}$ of attributes from $\mathfrak{A}$. Each attribute A of a relation schema is associated with a domain $dom(A)$ which represents the set of possible values that can occur in the column named A. If X and Y are sets of attributes, then we may write XY for $X \cup Y$.

If $X = \{A_1, \ldots, A_m\}$, then we may write $A_1 \cdots A_m$ for X. In particular, we may write simply A to represent the singleton $\{A\}$. A *tuple* over R (R-tuple or simply tuple, if R is understood) is a function $t : R \to \bigcup_{A \in R} dom(A)$ with $t(A) \in dom(A)$ for $i = 1, \ldots, n$. For $X \subseteq R$ let $t[X]$ denote the restriction of the tuple t over R on X, and $dom(X) = \prod_{A \in X} dom(A)$ the Cartesian product of the domains of attributes in X. A *relation* r over R is a finite set of tuples over R. Let $r[X] = \{t[X] \mid t \in r\}$ denote the *projection* of the relation r over R on $X \subseteq R$. For $X, Y \subseteq R$, $r_1 \subseteq dom(X)$ and $r_2 \subseteq dom(Y)$ let $r_1 \bowtie r_2 = \{t \in dom(XY) \mid \exists t_1 \in r_1, t_2 \in r_2 (t[X] = t_1[X] \wedge t[Y] = t_2[Y])\}$ be the *natural join* of r_1 and r_2.

Definition 1. *A* full first-order hierarchical decomposition *over the relation schema R is an expression $X : [Y_1 \mid \ldots \mid Y_k]$ with a non-negative integer k, $X, Y_1, \ldots, Y_k \subseteq R$ such that $Y_1, \ldots, Y_k$ forms a partition of $R - X$. A relation r over R is said to* satisfy *(or said to be a* model *of) the full first-order hierarchical decomposition $X : [Y_1 \mid \cdots \mid Y_k]$ over R, denoted by $\models_r X : [Y_1 \mid \cdots \mid Y_k]$, if and only if $r = (\cdots (r[XY_k] \bowtie r[XY_{k-1}]) \bowtie \cdots) \bowtie r[XY_1]$ holds.* □

The FOHD $\emptyset : [Y_1 \mid \cdots \mid Y_k]$ expresses the fact that any relation over R is the Cartesian product over its projections to attribute sets in $\{Y_i\}_{i=1}^k$. For $k = 0$, the FOHD $X : [\,]$ is satisfied trivially, where $[\,]$ denotes the empty list.

Suppose we allow the sets Y_i to be empty. Then for all positive k we have the property that for all relations r the FOHD $X : [\emptyset, Y_2, \ldots, Y_k]$ is satisfied by r if and only if r satisfies the FOHD $X : [Y_2, \ldots, Y_k]$. In particular, if $k = 1$, then $X : [\emptyset]$ is equivalent to $X : [\,]$; more specifically, they are satisfied by the same relations. One may now define an equivalence relation over the set of FOHDs defined on some fixed relation schema. Indeed, two such FOHDs are equivalent whenever they are satisfied by the same relations over the schema. Strictly speaking, we will apply inference rules to these equivalence classes of FOHDs. For the sake of simplicity, however, we have limited Definition 1 to those FOHDs where no empty sets are allowed to occur within the sequence of attribute sets. As the property above shows, this is not a real limitation but just a suitable choice of a representative from the equivalence classes.

For the design of a relational database schema semantic constraints are defined on the relations which are intended to be instances of the schema. During the design process one usually needs to determine further constraints which are logically implied by the given ones.

Definition 2. *Let $\Sigma \cup \{\varphi\}$ be a set of constraints on the relation schema R. We say that Σ* implies *φ if and only if every relation r over R that satisfies all constraints in Σ also satisfies φ.* □

In order to determine implied FOHDs one can use the set of inference rules from Table 2 [9,29]. These *inference rules* have the form

$$\frac{\text{premise}}{\text{conclusion}}$$

and inference rules without a premise are called *axioms*.

Table 2. Inference Rules for Full First-Order Hierarchical Decompositions

$$\frac{}{\emptyset : [R]}\ (\text{universal}, \mathcal{U})$$

$$\frac{X : [Y_1 \mid \cdots \mid Y_k]}{XZ : [Y_1 - Z \mid \cdots \mid Y_k - Z]}\ (\text{augmentation}, \mathcal{A})$$

$$\frac{X : [X_1 \mid X_2] \quad XX_i : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_1 \mid \cdots \mid Y_k \mid X_i]}\ (\text{transitivity}, \mathcal{T})$$

$$\frac{X : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_1 \mid \cdots \mid Y_i Y_j \mid \cdots \mid Y_k]}\ (\text{merging}, \mathcal{M})$$

$$\frac{X : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_{\pi(1)} \mid \cdots \mid Y_{\pi(k)}]}\ (\text{permutation}, \mathcal{P})$$

Let $\Sigma \cup \{\varphi\}$ be a set of semantic constraints from the class $\mathcal{C}$, all defined over the relation schema R. In this paper, the class $\mathcal{C}$ may refer to full first-order hierarchical dependencies or order-invariant hierarchical dependencies. Let $\Sigma \vdash_{\mathfrak{S}} \varphi$ denote the inference of φ from a set Σ of constraints in $\mathcal{C}$ by the set $\mathfrak{S}$ of inference rules. Let $\Sigma^+_{\mathfrak{S}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{S}} \varphi\}$ denote the *closure* of Σ under inferences by $\mathfrak{S}$. The set $\mathfrak{S}$ is called *sound* for the implication of constraints in $\mathcal{C}$ if for every relation schema R and for every set Σ of constraints in $\mathcal{C}$ over R we have $\Sigma^+_{\mathfrak{S}} \subseteq \Sigma^* = \{\varphi \in \mathcal{C} \mid \Sigma \text{ implies } \varphi\}$. The set $\mathfrak{S}$ is called *complete* for the implication of constraints in $\mathcal{C}$ if for every relation schema R and for every set Σ of constraints in $\mathcal{C}$ over R we have $\Sigma^* \subseteq \Sigma^+_{\mathfrak{S}}$. A complete set $\mathfrak{S}$ is called *minimal* for the implication of constraints in $\mathcal{C}$ if the removal of any inference rule from $\mathfrak{S}$ results in a system that is incomplete for the implication of constraints in $\mathcal{C}$.

Note the following global condition that we enforce on all applications of inference rules that infer FOHDs. Whenever we apply such an inference rule, we remove all empty sets from the exact position in which they occur as elements in the sequence in the conclusion. For instance, we can infer $R : [\,]$ by an application of the *augmentation rule* $\mathcal{A}$ to the FOHD $\emptyset : [R]$. The *split rule*

$$\frac{X : [X_1 \mid X_2] \quad X : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_1 \mid \cdots \mid Y_j \cap X_i \mid Y_j - X_i \mid \cdots \mid Y_k]}$$

is derivable from $\{\mathcal{A}, \mathcal{T}, \mathcal{M}, \mathcal{P}\}$.

Theorem 1. *The set $\mathfrak{F}$ of inference rules from Table 2 is sound, complete and minimal for the implication of full first-order decompositions.* □

The completeness argument in the proof of Theorem 1 is similar to the completeness proof for multivalued dependencies [3,5]. It relies on the notion of a *dependency basis*. Let $Dep_R(X)$ be the set of all $W \subseteq R - X$ for which some FOHD $X : [Y_1 \mid \cdots \mid Y_k]$ with $W \in \{Y_1, \ldots, Y_k\}$ can be inferred from Σ by $\mathfrak{F}$. Due to our definition of FOHDs, we must explicitly enforce the empty set to be

included in $Dep_R(X)$ as well. Note that $Dep_R(X)$ is finite, and $(Dep_R(X), \subseteq, \cup, \cap, (\cdot)^{\mathcal{C}}, \emptyset, R - X)$ constitutes a Boolean algebra due to the soundness of the merging and split rule. In particular, every finite Boolean algebra is atomic [16]. The set $DepB_R(X)$ of all atoms of $(Dep_R(X), \subseteq, \emptyset)$ is called the *dependency basis* of X with respect to Σ [2].

Even though $\mathfrak{F}$ forms a minimal axiomatisation one may still simplify the inference rules in $\mathfrak{F}$. For example, the transitivity rule $\mathcal{T}'$:

$$\frac{X : [X_1 \mid X_2] \quad XX_1 : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_1 \mid \cdots \mid Y_k \mid X_1]}$$

and the permutation rule $\mathcal{P}$ allow us to infer the transitivity rule $\mathcal{T}$. Moreover, the merging rule $\mathcal{M}'$:

$$\frac{X : [Y_1 \mid \cdots \mid Y_k]}{X : [Y_1 Y_2 \mid \cdots \mid Y_k]}$$

and the permutation rule $\mathcal{P}$ allow us to derive the merging rule $\mathcal{M}$.

Corollary 1. *The set $\mathfrak{F}' = \{\mathcal{U}, \mathcal{A}, \mathcal{T}', \mathcal{M}', \mathcal{P}\}$ is an axiomatisation for the implication of full first-order hierarchical decompositions.* □

3 The Role of the Permutation Rule

We will show now that the axiomatisation $\mathfrak{F}$ has the following property. For all relation schemata R, for all sets Σ of FOHDs on R, and for all inferences γ of an FOHD φ from Σ by $\mathfrak{F}$ there is an inference ξ of φ from Σ by $\mathfrak{F}$ in which the permutation rule is only applied in the very last step of ξ. Consequently, $\mathfrak{F}$ soundly reflects the role of the permutation rule as a means for fixing a decomposition strategy, but not as a means to derive elements of the dependency basis.

Example 5. The FOHD *Employee*:[*Insurance*|*Salary*, *Year*, *Child*] can be inferred by $\mathfrak{F}$ from *Employee*:[*Salary*, *Year*|*Child*,*Insurance*] and *Employee*:[*Child*| *Insurance*,*Salary*, *Year*]. Indeed, the inference

$$\dfrac{\dfrac{\dfrac{}{\mathcal{P}:\ Employee : [Salary,\ Year \mid Child,\ Insurance]}}{\mathcal{A}:\ Employee,\ Child : [Insurance \mid Salary,\ Year]}}{\mathcal{T}:\ Employee : [Insurance \mid Salary,\ Year \mid Child]}\ \mathcal{M}:\ Employee : [Insurance \mid Salary,\ Year,\ Child]$$

applies the permutation rule $\mathcal{P}$ in an intermediate step. According to our reasoning, this inference is not adequate. In fact,

$$\dfrac{\dfrac{\dfrac{}{\mathcal{A}:\ Employee,\ Child : [Salary,\ Year \mid Insurance]}}{\mathcal{T}:\ Employee : [Salary,\ Year \mid Insurance \mid Child]}}{\mathcal{M}:\ Employee : [Salary,\ Year,\ Child \mid Insurance]}\ \mathcal{P}:\ Employee : [Insurance \mid Salary,\ Year,\ Child]$$

shows an inference by $\mathfrak{F}$ which is indeed adequate. □

The reasoning in Example 5 applies to arbitrary inferences by $\mathfrak{F}$.

Theorem 2. *Let R be some relation schema, and Σ a set of FOHDs over R. For each inference γ from Σ by $\mathfrak{F}$ there is an inference ξ from Σ by $\mathfrak{F}$ with the following properties:*

- *γ and ξ infer the same full first-order hierarchical decomposition,*
- *in ξ the permutation rule $\mathcal{P}$ is applied at the very last step only.* □

Theorem 2 says that the set consisting of the universal axiom $\mathcal{U}$, the augmentation rule $\mathcal{A}$, the transitivity rule $\mathcal{T}$ and the merging rule $\mathcal{M}$ is *almost* complete for the implication of FOHDs in the following sense.

Corollary 2. *Let R be some relation schema and Σ a set of full first-order hierarchical decompositions on R. Then for all $X : [Y_1 \mid \cdots \mid Y_k]$ on R we have: $X : [Y_1 \mid \cdots \mid Y_k] \in \Sigma^+_{\mathfrak{F}}$ if and only if there is some permutation π on $\{1, \ldots, k\}$ such that $X : [Y_{\pi(1)} \mid \cdots \mid Y_{\pi(k)}] \in \Sigma^+_{\{\mathcal{U},\mathcal{A},\mathcal{T},\mathcal{M}\}}$.* □

Note that it is, by no means, self-evident that an axiomatisation of FOHDs has the property that applications of the permutation rule can always be deferred until the very last step of an inference. The axiomatisation $\mathcal{F}'$, for example, does not have this property.

4 Order-Invariant Hierarchical Dependencies

We will now introduce and study order-invariant hierarchical dependencies. A single order-invariant hierarchical dependency $X : \{Y_1, \ldots, Y_k\}$ is a compact representation of $k!$ many full first-order hierarchical decompositions.

Definition 3. *An* order-invariant hierarchical dependency *over relation schema R is an expression $X : \{Y_1, \ldots, Y_k\}$ with non-negative integer k, $X, Y_1, \ldots, Y_k \subseteq R$ such that $Y_1, \ldots, Y_k$ forms a partition of $R - X$. A relation r over R is said to* satisfy *the order-invariant hierarchical dependency $X : \{Y_1, \ldots, Y_k\}$ on R, denoted by $\models_r X : \{Y_1, \ldots, Y_k\}$, if and only if r is the natural join over $\{r[XY_i]\}_{i=1}^k$, i.e., if $r = r[XY_1] \bowtie \cdots \bowtie r[XY_k]$ holds.* □

Intuitively, the set-notation of OIHDs makes any application of the permutation rule unnecessary. The remarks regarding empty attribute subsets in FOHDs also apply to OIHDs. In particular, we apply the same global condition to inferences by the inference rules in Table 2.

Theorem 3. *The set $\mathfrak{O}$ of inference rules from Table 3 is sound, complete and minimal for the implication of OIHDs.* □

Corollary 2 enables us to reason about OIHDs by reasoning about FOHDs, and vice versa. Let φ denote the OIHD $X : \{Y_1, \ldots, Y_k\}$ if $\bar{\varphi}$ denotes the FOHD $X : [Y_1 \mid \cdots \mid Y_k]$, and let $\bar{\Sigma} = \{\bar{\sigma} \mid \sigma \in \Sigma\}$ denote the set of FOHDs if Σ denotes a set of OIHDs.

Table 3. Inference Rules for Order-Invariant Hierarchical Dependencies

$\dfrac{X : \{Y_1, \ldots, Y_k\}}{XZ : \{Y_1 - Z, \ldots, Y_k - Z\}}$ (augmentation, $\mathcal{A}$)	$\dfrac{X : \{X_1, X_2\} \quad XX_i : \{Y_1, \ldots, Y_k\}}{X : \{Y_1, \ldots, Y_k, X_i\}}$ (transitivity, $\mathcal{T}$)
$\dfrac{}{\emptyset : \{R\}}$ (universal axiom, $\mathcal{U}$)	$\dfrac{X : \{Y_1, \ldots, Y_k\}}{X : \{Y_1, \ldots, Y_iY_j, \ldots, Y_k\}}$ (merging, $\mathcal{M}$)

Corollary 3. *Let R be some relation schema, Σ a set of order-invariant hierarchical dependencies and $\bar{\Sigma}$ a set of full first-order hierarchical decompositions on R. Then for all OIHDs φ on R we have: $\varphi \in \Sigma^+_{\mathfrak{O}}$ if and only if $\bar{\varphi} \in \bar{\Sigma}^+_{\mathfrak{F}}$. Moreover, for all FOHDs $\bar{\varphi}$ on R we have: $\bar{\varphi} \in \bar{\Sigma}^+_{\mathfrak{F}}$ if and only if $\varphi \in \Sigma^+_{\mathfrak{O}}$.* □

5 Logic and Data Dependencies

Essentially, data dependencies are certain first-order formulae [13]. For instance, the OIHD $A : \{B, C, D\}$ over the relation schema R can be expressed by

$$\forall a, b, c, d, b', c', d', b'', c'', d''$$
$$((R(a, b, c, d) \wedge R(a, b', c', d') \wedge R(a, b'', c'', d'')) \Rightarrow R(a, b, c', d'')).$$

Binary OIHD $X : \{Y_1, Y_2\}$ are known as *multivalued dependencies* [12,5]. The implication of multivalued dependencies (MVDs) is even in one-to-one correspondence with fragments of propositional logic [10,11,27].

For a relation schema R let $\mathcal{V}_R = \{V_A : A \in R\}$ denote its corresponding set of propositional variables. For an OIHD $X : \{Y_1, \ldots, Y_k\}$ on R, denoted by φ, let φ' denote the following propositional formulae over $\mathcal{V}_R$:

$$\left(\bigwedge_{A \in X} V_A\right) \Rightarrow \left(\bigvee_{i=1}^{k} \left(\bigwedge_{B \in Y_i} V_B\right)\right). \tag{1}$$

For a set Σ of OIHDs over R let Σ' denote the set $\{\sigma' : \sigma \in \Sigma\}$ of propositional formulae over $\mathcal{V}_R$. The following theorem holds for MVDs [27].

Theorem 4. *[27] Let R be some relation schema, and let $\Sigma \cup \{\varphi\}$ be a set of MVDs over R. Then $\Sigma \models \varphi$ if and only if $\Sigma' \models \varphi'$.* □

However, for the class of OIHDs an extension of the correspondence to the fragment of formulae defined by Equation (1) fails. In fact, let us consider the OIHD $\sigma = A : \{\{B, C\}, \{D\}\}$ and the OIHD $\varphi = A : \{\{B\}, \{C\}, \{D\}\}$. We observe that σ does not imply φ as the two tuple relation $\{(a, b, c, d), (a, b', c', d)\}$ over $R = ABCD$ shows. However, $\sigma' = V_A \Rightarrow ((V_B \wedge V_C) \vee V_D)$ does logically imply $\varphi' = V_A \Rightarrow (V_B \vee V_C \vee V_D)$. Hence, the correspondence fails.

We will now briefly discuss which dependencies do correspond to the fragment of propositional logic defined by Equation (1). These are *degenerated OIHDs*, a subclass of Boolean dependencies [27]. The syntax of a *degenerated OIHD* is that of an OIHD. A relation r over R is said to *satisfy* the degenerated OIHD $X : \{Y_1, \ldots, Y_k\}$ over R if for all tuples $t_1, t_2 \in r$ the following holds: if $t_1[X] = t_2[X]$, then there is some $i \in \{1, \ldots, k\}$ such that $t_1[Y_i] = t_2[Y_i]$ holds. This is the same as saying that for all tuples $t_1, t_2 \in r$ it is true that for some $i \in \{1, \ldots, k\}$ the two-tuple relation $\{t_1, t_2\}$ satisfies the functional dependency $X \to Y_i$.

A consequence of Theorem 4 is that the implication of binary OIHDs corresponds exactly to the implication of degenerated OIHDs (i.e. degenerated multivalued dependencies) [27]. In general, however, the implication of OIHDs is not equivalent to the implication of degenerated OIHDs. For instance, if we view the OIHDs σ and φ as degenerated OIHDs, then σ does imply φ. Hence, σ implies φ when viewed as degenerated OIHDs, but σ does not imply φ when viewed as OIHD. Vice versa, φ implies σ when viewed as OIHDs, but φ does not imply σ when viewed as degenerated OIHDs.

6 Conclusion

We have established an axiomatisation of full first-order hierarchical decompositions that reflects the role of the permutation rule as a mere means to fix the order in which a database schema is decomposed. Hence, the permutation rule does not have any impact on the set of minimal units into which a database schema can be separated, i.e., the elements of the dependency basis. Indeed, our axiomatisation allows applications of the permutation rule to be delayed until the very last step of an inference. Removing the permutation rule from the axiomatisation results in a set of inference rules that is sound and complete for the implication of order-invariant hierarchical dependencies.

The results of this paper are complementary to Biskup's results on the role of the complementation rule in the context of multivalued dependencies [4]. Indeed, there are axiomatisations of MVDs in which applications of the complementation rule could be avoided completely or deferred until the very last step of an inference [4,25]. Consequently, the complementation rule is a mere means to achieve database normalisation. These results were extended to the context of partial database relations [26], full hierarchical dependencies [22], functional and multivalued dependencies [5].

The eXtensible Markup Language (XML) [6] offers a flexible way to store data. As such, it has received considerable interest from the database community. One challenging area of XML research addresses constraints, which provide effective means to capture important semantic information about XML data [15]. So far, the interest in XML constraints has mainly addressed keys [7,18,20,21] and functional dependencies [1,17,23,31,32]. As far as the author is aware, the decomposition of XML data based on integrity constraints has not been studied.

References

1. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Trans. Database Syst. 29(1), 195–232 (2004)
2. Beeri, C.: On the membership problem for functional and multivalued dependencies in relational databases. ACM Trans. Database Syst. 5(3), 241–259 (1980)
3. Beeri, C., Fagin, R., Howard, J.H.: A complete axiomatization for functional and multivalued dependencies in database relations. In: SIGMOD, pp. 47–61. ACM Press, New York (1977)
4. Biskup, J.: Inferences of multivalued dependencies in fixed and undetermined universes. Theor. Comput. Sci. 10(1), 93–106 (1980)
5. Biskup, J., Link, S.: Appropriate reasoning about data dependencies in fixed and undetermined universes. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 58–77. Springer, Heidelberg (2008)
6. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible markup language (XML) 1.0, 4th edn. W3C recommendation (2006), http://www.w3.org/TR/xml
7. Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W.: Keys for XML. Computer Networks 39(5), 473–487 (2002)
8. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
9. Delobel, C.: Normalisation and hierarchical dependencies in the relational data model. ACM Trans. Database Syst. 3(3), 201–222 (1978)
10. Demetrovics, J., Rónyai, L., Son, H.: On the representation of dependencies by propositional logic. In: Thalheim, B., Gerhardt, H.-D., Demetrovics, J. (eds.) MFDBS 1991. LNCS, vol. 495, pp. 230–242. Springer, Heidelberg (1991)
11. Fagin, R.: Functional dependencies in a relational data base and propositional logic. IBM Journal of Research and Development 21(6), 543–544 (1977)
12. Fagin, R.: Multivalued dependencies and a new normal form for relational databases. ACM Trans. Database Syst. 2(3), 262–278 (1977)
13. Fagin, R.: Horn clauses and database dependencies. J. ACM 29(4), 952–985 (1982)
14. Fagin, R., Vardi, M.Y.: The theory of data dependencies: a survey. In: Mathematics of Information Processing: Proceedings of Symposia in Applied Mathematics, pp. 19–71. American Mathematical Society (1986)
15. Fan, W.: XML constraints. In: DEXA Workshops 2005: Proceedings of the 16th International Workshop on Database and Expert Systems Applications, pp. 805–809. IEEE Computer Society Press, Los Alamitos (2005)
16. Graetzer, G.: General Lattice Theory. Birkhäuser, Basel (1998)
17. Hartmann, S., Link, S.: More functional dependencies for XML. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 355–369. Springer, Heidelberg (2003)
18. Hartmann, S., Link, S.: Numerical keys for XML. In: Leivant, D., de Queiroz, R. (eds.) WoLLIC 2007. LNCS, vol. 4576, pp. 203–217. Springer, Heidelberg (2007)
19. Hartmann, S., Link, S.: Characterising nested database dependencies by fragments of propositional logic. Ann. Pure Appl. Logic 152(1-3), 84–106 (2008)
20. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2:2) (2009)
21. Hartmann, S., Link, S.: Expressive, yet tractable XML keys. In: EDBT, ACM International Conference Proceeding Series, vol. (360), pp. 357–367 (2009)

22. Hartmann, S., Link, S., Köhler, H.: Full hierarchical dependencies in fixed and undetermined universes. Ann. Math. Artif. Intell. 50(1-2), 195–226 (2007)
23. Hartmann, S., Trinh, T.: Axiomatising functional dependencies for XML with frequencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 159–178. Springer, Heidelberg (2006)
24. Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2006)
25. Link, S.: Charting the completeness frontier of inference systems for multivalued dependencies. Acta Inf. 45(7-8), 565–591 (2008)
26. Link, S.: On the implication of multivalued dependencies in partial database relations. Int. J. Found. Comput. Sci. 19(3), 691–715 (2008)
27. Sagiv, Y., Delobel, C., Parker Jr., D.S., Fagin, R.: An equivalence between relational database dependencies and a fragment of propositional logic. J. ACM 28(3), 435–453 (1981)
28. Thalheim, B.: Dependencies in Relational Databases. Teubner-Verlag (1991)
29. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Heidelberg (2000)
30. Vincent, M.: Semantic foundation of 4NF in relational database design. Acta Inf. 36, 1–41 (1999)
31. Vincent, M., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM Trans. Database Syst. 29(3), 445–462 (2004)
32. Yu, C., Jagadish, H.: XML schema refinement through redundancy detection and normalization. VLDB J. 17(2), 203–223 (2008)

Classic-Like Analytic Tableaux for Finite-Valued Logics

Carlos Caleiro[1] and João Marcos[2]

[1] Instituto de Telecomunicações and Dept. Mathematics, IST, TU-Lisbon, Portugal
[2] Department of Informatics and Applied Mathematics, UFRN, Brazil

Abstract. The paper provides a recipe for adequately representing a very inclusive class of finite-valued logics by way of tableaux. The only requisite for applying the method is that the object logic received as input should be sufficiently expressive, in having the appropriate linguistic resources that allow for a bivalent representation. For each logic, the tableau system obtained as output has some attractive features: exactly two signs are used as labels in the rules, as in the case of classical logic, providing thus a uniform framework in which different logics can be represented and compared; the application of the rules is analytic, in that it always reduces complexity, providing thus an immediate proof-theoretical decision procedure together with a counter-model builder for the given logic.

Keywords: many-valued logics, proof theory.

1 Background

The fact that any abstract consequence relation may be represented by way of an adequate many-valued semantics (cf. [16]) makes many-valued logics ubiquitous in the realm of inference systems. Moreover, the compositional feature that characterizes truth-functional semantics makes the latter extremely attractive for computational or linguistic purposes. This much from a purely semantical perspective. From a proof-theoretical perspective, on the other hand, the existence of appropriate and efficient deductive formalisms and theorem-proving frameworks for truth-functional logics provides many-valued logics with useful tools for automating their variegated approaches to entailment and for developing deep computational insights into their underlying reasoning mechanisms.

General procedures for providing arbitrary finite-valued logics with adequate tableau systems are known since long in the literature (cf. [3,8]). Reasonably up-to-date implementation-oriented accounts of such axiom-extraction strategies can be found in [10,4]. The price to pay for the full generality of such approaches is that of a certain semantic intromission in the proof-theoretical formulation of the corresponding object logics: the tableau rules, in each case, contain formulas labeled by as many different signs as there are truth-values, or collections of truth-values. The issue here goes beyond a mere sacrifice in elegance: the staggering and heavily semantic-dependent wealth of signs for formulas culminates

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 268–280, 2009.

in irksome difficulties for the task of comparing different logics on what regards their deductive strengths, given the inexistence of a uniform object-language framework for dealing with all of them at once.

The intrinsic bivalence that underlies the usual definition of entailment for many-valued logics has suggested that the many 'algebraic' truth-values of the latter might be represented by the use of just two 'logical' values (cf. [15]). A constructive procedure for producing an equivalent bivalent semantics for any sufficiently expressive finite-valued logic has been proposed in [6]. Suitable machineries for extracting classic-like sequent systems for generous classes of such bivalent semantics were set up in [5,1], and a sketch of how any such bivalent semantics may give origin to classic-like 2-signed tableaux was offered in [6] and implemented in [13]. The advantage of uniformity of framework provided by the mentioned constructive extraction of adequate 2-signed tableau-theoretic formalizations for finite-valued logics was partially canceled, however, by the fact that among such tableau rules a non-analytic dual branching version of the cut rule was to be found. In contrast, the present paper is to show in detail how adequate classic-like tableau systems may be constructively extracted directly from the corresponding finite-valued semantics, this time with the additional advantage of analyticity, a feature that allows for the immediate design of fully automated decision tacticals for the logics characterized by such semantics.

2 Truth-Functionality *vs.* Bivalence

Consider an alphabet consisting of a denumerable set $\mathcal{A} = \{p_0, p_1, p_2, \ldots\}$ of *atoms/variables* and a finite set of *connectives* $\Sigma = \{\odot_0, \odot_1, \ldots, \odot_k\}$. The arity of a given connective $\odot \in \Sigma$ will be denoted by $\mathsf{ar}\odot$. The set $\mathbb{S}$ of *formulas*, as usual, is the algebra freely generated over $\mathcal{A}$ with respect to Σ. Let $\mathcal{V}_n = \{v_0, v_1, \ldots, v_{n-1}\}$ be a set of *truth-values*, partitioned into a set $\mathcal{D} \subseteq \mathcal{V}_n$ of *designated* values and a set $\mathcal{U} = \mathcal{V}_n \setminus \mathcal{D}$ of *undesignated* values. In what follows, it will be handy in many cases to assume $F = v_0$ and $T = v_{n-1}$. In general, an n-(*valued*) *assignment* of truth-values to the atoms is any mapping $\rho : \mathcal{A} \to \mathcal{V}_n$, and an n-(*valued*) *valuation* is any extension $w : \mathbb{S} \to \mathcal{V}_n$ of such an assignment to the set of all formulas. An *n-valent semantics* for $\mathbb{S}$ based on $\mathcal{V}_n$, then, is simply an arbitrary collection of n-valued valuations. In particular, we will call *bivalent* any (classic-like) semantics where $\mathcal{V}_2 = \{F, T\}$ and $\mathcal{D}_2 = \{T\}$; the corresponding valuations are called *bivaluations*. Canonical notions of *entailment* $\models_x \subseteq \mathsf{Pow}(\mathbb{S}) \times \mathbb{S}$ characterizing a *logic* $\mathcal{L}$ may be associated to any valuation w and any n-valent semantics Sem, if one simply sets $\Gamma \models_w \alpha$ iff $(w(\alpha) \in \mathcal{D}$ whenever $w(\Gamma) \subseteq \mathcal{D})$, and $\Gamma \models_{\mathsf{Sem}} \alpha$ iff $(\Gamma \models_w \alpha$ for every $w \in \mathsf{Sem})$, where $\Gamma \cup \{\alpha\} \subseteq \mathbb{S}$. Any pair $\langle \Gamma, \alpha \rangle \in \mathsf{Pow}(\mathbb{S}) \times \mathbb{S}$ such that $\Gamma \models_{\mathsf{Sem}} \alpha$ is called a *valid inference* of Sem.

A particularly interesting case of n-valent semantics corresponds to the ones we call *truth-functional*, namely, semantics provided to the set of formulas $\mathbb{S}$ by defining an appropriate Σ-algebra $\mathbb{V}$ with carrier $\mathcal{V}_n$, associating to each $\odot \in \Sigma$ an $\mathsf{ar}\odot$-ary operator $\widehat{\odot} \in \mathbb{V}$, and collecting in Sem the set of all homomorphisms $\S : \mathbb{S} \to \mathbb{V}$. Any such homomorphism, as is usual in the field of

universal algebra, can be understood as the unique extension of an assignment $\rho : \mathcal{A} \to \mathcal{V}_n$ into a valuation $\S_\rho : \mathbb{S} \to \mathbb{V}$ where one imposes $\S(\odot(\varphi_1, \ldots, \varphi_{\mathsf{ar}_\odot})) = \widehat{\odot}(\S(\varphi_1), \ldots, \S(\varphi_{\mathsf{ar}_\odot}))$. This way, one might say that such a semantics is *compositional*, in that the meaning it attributes to a complex expression clearly depends (functionally) on the meaning of its directly subordinated subexpressions. Any logic characterized by truth-functional means, for a given $\mathcal{V}_n$, is called *n-valued*; we will say that an n-valued logic $\mathcal{L}$ with an entailment relation $\models_{\mathsf{Sem}}$ is *genuinely n-valued* in case there is no $m < n$ such that $\models_{\mathsf{Sem}}$ can be canonically obtained by way of an m-valued truth-functional semantics.

Example 1. Our running example for this paper will involve a well-known class of truth-functional n-valued logics, namely Łukasiewicz's logics $\mathrm{Ł}_n$, for $n > 2$. Each $\mathrm{Ł}_n$ may be characterized by considering the unary connective $\neg$ and the binary connective $\supset$, together with a set of truth-values $\mathcal{V}_n$ where the designated ones form the singleton $\mathcal{D} = \{v_{n-1}\}$, while interpreting $\widehat{\neg} v_i$ as $v_{(n-1)-i}$ and interpreting $v_i \widehat{\supset} v_j$ as $v_{(n-1)-(i-j)}$ in case $i > j$, and as $v_{(n-1)}$ otherwise, where $0 \leq v_i, v_j \leq n-1$.

Taking advantage of the residual shadow of bivalence that lurks in the distinction between designated and undesignated truth-values, it is easy to see that any entailment relation characterizing an n-valued logic can also be characterized by way of a bivalent semantics. Indeed, consider the total mapping $t : \mathcal{V}_n \to \mathcal{V}_2$ such that $t(v) = T$ iff $v \in \mathcal{D}$ and define, for any valuation $\S : \mathbb{S} \to \mathbb{V}$ of an n-valued semantics Sem, the bivaluation $b_\S = t \circ \S$. Collect all such bivaluations into a semantics Sem_2 and notice that $\Gamma \models_x \alpha$ iff $\Gamma \models_y \alpha$, where $\langle x, y\rangle \in \{\langle \S, b_\S\rangle, \langle \mathsf{Sem}, \mathsf{Sem}_2\rangle\}$.

Now, given a genuinely n-valued logic, for $n > 2$, describing this same logic by way of a bivalent (non-truth-functional) semantics would seem to throw away the fundamental feature of compositionality, making the resulting semantic characterization less appealing both from a meta-theoretical and from a practical point of view. As we will see in what follows, however, this is not necessarily the case, as bivalent semantics can be quite profitable and informative, even in the non-truth-functional case, where in fact an extended notion of compositionality may be entertained.

Let's first try and find a way of distinguishing each pair of values of a genuinely n-valued logic $\mathcal{L}$. Given $v_i, v_j \in \mathcal{V}$, we write $v_i \sharp v_j$ and say that v_i and v_j are *separated* in case v_i and v_j belong to different classes of truth-values, that is, in case $t(v_i) \neq t(v_j)$. Given any two variables p_i and p_j and any valuation $\S$ such that $v_i = \S(p_i) \neq \S(p_j) = v_j$ yet $b_\S(p_i) = b_\S(p_j)$, we say that a one-variable formula $\theta^{ij}(p)$ of $\mathcal{L}$ *separates* v_i and v_j if $\S(\theta^{ij}(p_i)) \sharp \S(\theta^{ij}(p_j))$ (or, equivalently, $b_\S(\theta^{ij}(p_i)) \neq b_\S(\theta^{ij}(p_j))$). In that case we will also say that the values v_i and v_j of $\mathcal{L}$ are *effectively distinguishable*, as they may be separated using just the original linguistic resources of $\mathcal{L}$. Finally, we will say that the logic $\mathcal{L}$ is *effectively separable* in case its truth-values are pairwise effectively distinguishable, that is, for any pair of distinct values $\langle v_i, v_j\rangle \in \mathcal{D}^2 \cup \mathcal{U}^2$ a one-variable formula $\theta^{ij}(p)$ can be found in $\mathcal{L}$ that separates v_i and v_j. Collect, without repetition, all such one-variable formulas into a finite sequence $\theta_1(p), \ldots, \theta_s(p)$, and assume $\theta_0(p) = p$;

obviously, $\theta_0(p)$ by itself suffices to separate any pair of values $\langle v_i, v_j\rangle \in (\mathcal{D} \times \mathcal{U}) \cup (\mathcal{U} \times \mathcal{D})$. Then, the *binary print* of a value $v \in \mathcal{V}$ will be the sequence $\overline{v} = [b_{\S}(\theta_r(p))]_{r=0}^{s}$, where $\S(p) = v$. Notice that for every pair of distinct values $\langle v_i, v_j\rangle \in \mathcal{V}^2$ it is now obviously the case that $\overline{v_i} \neq \overline{v_j}$.

Example 2. Back to the example of the Ł$_n$, one has to devise a way of pairwise separating each of the $n-1$ undesignated values, in each case. A well-known general method, in the case of the Ł$_n$, is to use the Rosser-Turquette functions (cf. [14]). To give an independent illustration of how the separation can be done in the particular case of Ł$_3$, one might either define $\theta^{01}(p) = \theta_1(p)$ as $\neg p$, using a primitive connective of the language of Ł$_3$, or alternatively define this same $\theta_1(p)$ using a more complex formula such as $\neg p \supset p$. To simplify notation, in this particular case of Ł$_3$, where a single separating formula θ_1 suffices, we shall drop its subscript. For different reasons, it is obvious in each case that $\overline{v_0} \neq \overline{v_1}$. Indeed, using the first definition, the binary prints corresponding respectively to v_0, v_1 and v_2 are $\langle F, T\rangle$, $\langle F, F\rangle$ and $\langle T, F\rangle$; the second definition originates, respectively, the binary prints $\langle F, F\rangle$, $\langle F, T\rangle$ and $\langle T, T\rangle$.

The next sections will show how such effective separation of truth-values, whenever it can be effected —and that is a decidable property of a finite-valued logic—, may be used to automatically produce adequate classic-like analytic tableau systems for the corresponding finite-valued logics.

3 A Uniform Analytic Deductive Formalism

The result, mentioned in the last section, that allowed for the characterization of a finite-valued logic by way of bivalent semantics, coupled with the technique that allows for the separation of the algebraic truth-values of the object logic by way of the binary print defined with the help of the linguistic resources of that very logic, gives a hint on how the corresponding adequate bivalent semantics may be constructively described, in each case. A further step will now be to show in detail how this bivalence may be explored in order to devise an adequate classic-like formalism to investigate a finite-valued logic from a proof-theoretic perspective. Before outlining the general method we intend to propose, having as output an appropriate labeled (in fact, 2-signed) tableau system for some given sufficiently expressive finite-valued logic, let's illustrate it in the present section with a fully worked example. We shall use & to represent conjunction in the classical metalanguage, || to represent disjunction, $\Longrightarrow$ to represent implication, and $\divideontimes$ to represent an absurd.

Now, consider again the case of Ł$_3$, where $\Sigma = \{\neg, \supset\}$, and recall the particular separation of truth-values produced by setting $\theta(p) = \neg p \supset p$. It follows that $\theta(\neg\varphi_1) = \neg\neg\varphi_1 \supset \neg\varphi_1$ and $\theta(\varphi_1 \supset \varphi_2) = \neg(\varphi_1 \supset \varphi_2) \supset (\varphi_1 \supset \varphi_2)$. Using the 3-valued semantics of Ł$_3$ one will then notice that:

$\S(\theta(\neg\varphi_1)) = v_0$ only if $\S(\varphi_1) = v_2$ and $\S(\theta(\neg\varphi_1)) = v_2$ only if $\S(\varphi_1) \in \{v_0, v_1\}$

Recalling the binary prints of v_0 as $\langle F, F\rangle$, of v_1 as $\langle F, T\rangle$, and of v_2 as $\langle T, T\rangle$, one might rewrite now the above by way of the following first-order schematic sentences, whose consequents are written in a kind of 'disjunctive normal form':

(Ł_3.1) $F{:}\theta(\neg\varphi_1) \Longrightarrow (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1))$
(Ł_3.2) $T{:}\theta(\neg\varphi_1) \Longrightarrow (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1)) \;||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1))$

Similarly, one might also notice that:

$\S(\theta(\varphi_1 \supset \varphi_2)) = v_0$ only if $\S(\varphi_1) = v_2$ and $\S(\varphi_2) = v_0$,
$\S(\theta(\varphi_1 \supset \varphi_2)) \neq v_1$ for every valuation $\S$, and
$\S(\theta(\varphi_1 \supset \varphi_2)) = v_2$, otherwise.

These immediately translate into:

(Ł_3.3) $F{:}\theta(\varphi_1 \supset \varphi_2) \Longrightarrow (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2))$
(Ł_3.4) $T{:}\theta(\varphi_1 \supset \varphi_2) \Longrightarrow (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$

We will call the expressions (Ł_3.1)–(Ł_3.4) *θ-rules*. The full description of Ł_3 will also include the following *non-θ-rules*, obtained by using the binary prints now to describe the original truth-tables of the primitive connectives of Ł_3:

(Ł_3.5) $F{:}\neg\varphi_1 \Longrightarrow (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1)) \;||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1))$
(Ł_3.6) $T{:}\neg\varphi_1 \Longrightarrow (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1))$
(Ł_3.7) $F{:}(\varphi_1 \supset \varphi_2) \Longrightarrow (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2))$
$||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2)) \;||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2))$
(Ł_3.8) $T{:}(\varphi_1 \supset \varphi_2) \Longrightarrow$
$(F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; F{:}\theta(\varphi_2)) \;||\; (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; F{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2)) \;||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; F{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$
$||\; (F{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2)) \;||\; (T{:}\varphi_1 \;\&\; T{:}\theta(\varphi_1) \;\&\; T{:}\varphi_2 \;\&\; T{:}\theta(\varphi_2))$

The above first-order expressions are intended to represent tableau rules: the antecedent of each rule is the head, and the disjunction in the consequent describes the branches that may be created once the head is matched on a previously given branch. In addition to the traditional closure rule for 2-signed tableaux, which says that a branch is closed once it contains two signed formulas of the form $F{:}\varphi$ and $T{:}\varphi$, additional closure rules will be needed in order to exclude each binary print not allowed for the given logic with a given choice of separating formulas. In the present case, as only the pair $\langle T, F\rangle$ fails to be among the pairs allowed as binary prints of the initial collection of truth-values, an additional closure rule will say that branches containing both a signed formula of the form $T{:}\varphi$ and a signed formula of the form $F{:}\theta(\varphi)$ may be closed. One might represent such closure rules by writing:

(Ł_3.C0) $F{:}\varphi \;\&\; T{:}\varphi \Longrightarrow \divideontimes$ (Ł_3.C1) $T{:}\varphi \;\&\; F{:}\theta(\varphi) \Longrightarrow \divideontimes$

There is, of course, a lot of redundancy to be found among the above mechanically extracted θ-rules and non-θ-rules. A simpler tableau system for Ł$_3$ might be defined, though, by any set of expressions first-order-equivalent to (Ł$_3$.1)–(Ł$_3$.8), together with the already mentioned appropriate closure rules. An example of such a simpler axiomatization is provided by (Ł$_3$.C0) and (Ł$_3$.C1) together with:

θ-rules

$(\text{Ł}_3.1)^*\ F{:}\theta(\neg\varphi_1) \Longrightarrow T{:}\varphi_1 \qquad (\text{Ł}_3.2)^*\ T{:}\theta(\neg\varphi_1) \Longrightarrow F{:}\varphi_1$

$(\text{Ł}_3.3)^*\ F{:}\theta(\varphi_1 \supset \varphi_2) \Longrightarrow (T{:}\varphi_1\ \&\ F{:}\theta(\varphi_2))$

$(\text{Ł}_3.4)^*\ T{:}\theta(\varphi_1 \supset \varphi_2) \Longrightarrow F{:}\varphi_1\ \mid\mid\ T{:}\theta(\varphi_2)$

non-θ-rules

$(\text{Ł}_3.5)^*\ F{:}\neg\varphi_1 \Longrightarrow T{:}\theta(\varphi_1) \qquad (\text{Ł}_3.6)^*\ T{:}\neg\varphi_1 \Longrightarrow F{:}\theta(\varphi_1)$

$(\text{Ł}_3.7)^*\ F{:}(\varphi_1 \supset \varphi_2) \Longrightarrow (T{:}\varphi_1\ \&\ F{:}\varphi_2)\ \mid\mid\ (T{:}\theta(\varphi_1)\ \&\ F{:}\theta(\varphi_2))$

$(\text{Ł}_3.8)^*\ T{:}(\varphi_1 \supset \varphi_2) \Longrightarrow F{:}\theta(\varphi_1)\ \mid\mid\ T{:}\varphi_2\ \mid\mid\ (F{:}\varphi_1\ \&\ T{:}\theta(\varphi_2))$

Figure 1 (see Appendix) shows an example of a tableau for Ł$_3$, using the above simpler set of rules. There, the branches (2.1) and (2.2) originate from the application of rule (Ł$_3$.7)* to the signed formula (1), and the same rule applies to (2.1.2) to originate branches (3.1) and (3.2). The branch that goes through (4) originates from the application of rule (Ł$_3$.3)* to the signed formula (2.2.2). The usual closure rule for tableaux, (Ł$_3$.C0), closes the leftmost branch, in view of the nodes (2.1.1) and (3.1.2). Similarly for the rightmost branch, in view of (2.2.1) and (4.2), even though the involved formula in this case is non-atomic. As for the innermost branch, notice that (3.2.2) has the form $F{:}\theta(p_0)$, so the additional closure rule of Ł$_3$, (Ł$_3$.C1), finishes off the branch, in view of (2.1.1).

Special care must be exercised, in the present environment, as θ-rules might interfere with non-θ-rules, or with other θ-rules, and as the blind application of tableau rules might make the same signed formula appear over and over again, in the same branch. Even worse, it would appear that, applied in the wrong order, some signed formulas might give rise to increasingly more complex signed formulas. For instance, suppose that in the tableau from Fig. 1 one never applied the combination of rules (Ł$_3$.3)* and (Ł$_3$C0) after (2.2.2), but applied instead rule (Ł$_3$.8)* to the signed formula $T{:}\theta(p_0)$ that appears in (2.2.1). One of the originating branches would then extend this branch exactly by adding the signed formula $F{:}\theta(\neg p_0)$. If, further on, to this new signed formula one applied rule (Ł$_3$.8)* instead of rule (Ł$_3$.1)*, a new branch would originate in which the signed formula $T{:}\theta(\neg\neg p_0)$ were added. Such unfortunate sequential choice of rules could of course go on forever, producing more and more complex signed formulas, originating thus a tableau branch that would never be closed.

A way of avoiding the above phenomenon would consist in only allowing rule application in building tableaux when the signed formulas originating from a given node are strictly 'less complex' than the signed formula present in that node. One might realize such intent by choosing an appropriate *complexity measure* $\ell : \mathbb{S} \to \mathbb{N}$ for formulas that is guaranteed to decrease with rule application. In the particular case of Ł$_3$ one might define $\ell(p) = 0$, $\ell(\theta(\varphi)) = \ell(\varphi)$, $\ell(\neg\varphi_1) = \ell(\varphi_1) + 1$ and $\ell(\varphi_2 \supset \varphi_3) = \ell(\varphi_2) + \ell(\varphi_3) + 1$, where $p \in \mathcal{A}$, $\varphi, \varphi_1, \varphi_2, \varphi_3 \in \mathbb{S}$, and where φ_2 is not of the form $\neg\varphi_3$, that is, $\varphi_2 \supset \varphi_3$ does not

appear at the head of a θ-rule. One might now use such complexity measure as a guide while constructing tableaux for Ł$_3$, observing that: (i) no rule applies to an atom p, and similarly no rule should be applied to a formula of the form $\theta(p)$, as both p and $\theta(p)$ have complexity zero; (ii) θ-rules contribute more to the reduction of complexity than non-θ-rules, as a formula of the form $\theta(\varphi)$ has the same complexity as φ, and the consequent of an application of a θ-rule involves only formulas of lower complexity, of the forms φ_r or $\theta(\varphi_r)$, where φ_r is a proper subformula of φ. Furthermore, although it is not the case for Ł$_3$, it may happen in general that more than one θ-rule is applicable to the same signed formula. In such a case, as we will impose below, one ought to choose the θ-rule whose head is more 'concrete'. As it turns out, it is always possible to order the rules in a way that solves all ambiguities while guaranteeing also that all tableau constructions terminate. We shall dub such tableau-building heuristics a *requirement of analyticity*, and, as we shall see, it will guarantee that our tableau proofs are normalized and terminate. In the case of the previous example, the requirement of analyticity would guarantee that rule (Ł$_3$.8)* would never be applied to node (2.2.1), as the formula with sign T that appears in the that node already has complexity zero. Indeed, as we have seen, modifying the above example in allowing a non-θ-rule to be applied before a θ-rule in a situation involving non-atomic formulas turned out to allow for the production of increasingly more complex formulas — as it is obviously the case, for instance, that $\ell(\theta(\varphi)) < \ell(\theta(\neg\varphi))$.

The next section will show how the above illustrated procedure may be generalized so as to provide adequate and well-behaved proof-theoretical formalisms, together with a classic-like decision procedure, for a very inclusive class of truth-functional logics.

4 The Extraction of Adequate Classic-Like Tableau Systems for Finite-Valued Logics

Let $\mathcal{L}$ be an effectively separable n-valued logic with a set of formulas $\mathbb{S}$ generated over the denumerable set of atoms $\mathcal{A} = \{p_0, p_1, p_2, \ldots\}$ with respect to the set of connectives $\Sigma = \{\odot_0, \odot_1, \ldots, \odot_k\}$ and having $\mathcal{D} \subseteq \mathcal{V}_n$ as its set of designated values, and assume that its binary prints are produced by a convenient sequence of one-variable separating formulas $\theta_1(p), \ldots, \theta_s(p)$. Recall that we set $\theta_0(p) = p$. So, for each truth-value $v \in \mathcal{V}_n$, we might take an atom p and an n-valued assignment $\S$ such that $\S(p) = v$, and consider the distinctive characterizing bivalent sequence $\overline{v} = [b_\S(\theta_r(p))]_{r=0}^s$. As a matter of convention, we shall say that an n-valued valuation $\S$ *satisfies* a labeled formula of the form $X{:}\delta$ if $b_\S(\delta) = X$.

As for the associated tableau rules, consider first the usual classic-like closure rule (C0) of the form: $F{:}\varphi$ & $T{:}\varphi \Longrightarrow \divideontimes$. Furthermore, let $\mathsf{BP} = \{[c_r]_{r=0}^s : c_r \in \{F, T\}\}$ be the set of all possible $(s+1)$-long binary prints, and let $\mathsf{CL} = \mathsf{BP} \setminus \{\overline{v} : v \in \mathcal{V}_n\}$ be the set of all such bivalent sequences that are *not* produced as binary prints of truth-values of $\mathcal{L}$. Intuitively, any *closuring sequence* $\widetilde{c} \in \mathsf{CL}$ brings about information that is unobtainable by way of the initial truth-values of $\mathcal{L}$, allowing one thus to close a tableau branch that contains such a sequence.

Information, even if partial, leading unambiguously to a binary print in CL should always give rise to a closed tableau. Let a *partial binary print* be any sequence $\widetilde{c}_R = [c_r]_{r\in R}$ such that $R \subseteq \{0,\ldots,s\}$ and each $c_r \in \{F,T\}$ (this definition includes, of course, the total binary prints in BP, as strict partiality occurs exactly when R is a proper subset of $\{0,\ldots,s\}$). Given two partial binary prints $\widetilde{c}_{R_1}$ and $\widetilde{d}_{R_2}$, we say that $\widetilde{c}_{R_1}$ extends $\widetilde{d}_{R_2}$ if $R_2 \subsetneq R_1$ and $d_r = c_r$ for every $r \in R_2$. We can now conclude that closuring information is carried by any partial binary print $\widetilde{c}_R$ such that all of its $2^{s+1-\mathsf{Card}(R)}$ possible total extensions are in CL. Hence, it would be reasonable to add a different closure rule for each such partial closuring information. However, it suffices to take into account just the minimal closuring situations, that is, closuring partial binary prints $\widetilde{c}_R$ that cannot be obtained as extensions of any other closuring partial binary print.

In general, where $\widetilde{c}_R = [c_r]_{r\in R}$ is some partial binary print, and δ stands for an arbitrary schematic formula, we write $\widetilde{c}^{\mathbb{S}}_R(\delta) = [c_r{:}\theta_r(\delta)]_{r\in R}$ for the linguistic 2-signed version of such partial binary print. Accordingly, for each minimal closuring partial binary print $\widetilde{c}_R$, consider an additional closure rule (C#) of the form: $\&(\widetilde{c}^{\mathbb{S}}_R(\varphi)) \Longrightarrow *$.

Recall, moreover, that for each connective $\odot : \mathbb{S}^j \to \mathbb{S}$ of Σ with arity $j = \mathsf{ar}\odot$ there is an associated operator $\widehat{\odot} : \mathcal{V}_n^j \to \mathcal{V}_n$ in the algebra of truth-values. This interpretation mapping can immediately be extended homomorphically to any formula δ of any given arity, and we shall denote this by $\widehat{\delta}$. Finally, consider again the flattening total mapping $t : \mathcal{V}_n \to \{F,T\}$ such that $t(v) = T$ iff $v \in \mathcal{D}$. Given $X \in \{F,T\}$, a j-ary connective $\odot$ and a separating formula θ, let $B^{\theta\odot}_X([\varphi_i]_{i=1}^j) = \{\&[\overline{v_i}^{\mathbb{S}}(\varphi_i)]_{i=1}^j : t(\widehat{\theta}(\widehat{\odot}([v_i]_{i=1}^j))) = X\}$. For each $B^{\theta\odot}_X([\varphi_i]_{i=1}^j)$ consider then a rule of the form: $X{:}\theta(\odot([\varphi_i]_{i=1}^j)) \Longrightarrow \mid\mid B^{\theta\odot}_X([\varphi_i]_{i=1}^j)$. Such rules are called *θ-rules* when $\theta = \theta_i$, for some $0 < i \leq s$; otherwise, when $\theta = \theta_0(p)$, they are called *non-θ-rules*. Notice that those rules generate as many branches as there are members (conjunctions) of $B^{\theta\odot}_X([\varphi_i]_{i=1}^j)$. The number of different non-θ-rules is $2\times\mathsf{Card}(\Sigma)$, and there are $2\times s\times\mathsf{Card}(\Sigma)$ different θ-rules. For each fixed j-ary connective $\odot$ and each fixed separating formula θ, the summed number of branches of the rules $B^{\theta\odot}_F([\varphi_i]_{i=1}^j)$ and $B^{\theta\odot}_T([\varphi_i]_{i=1}^j)$ generated by the above procedure always amounts to n^j; additionally, each branch will have exactly $s+1$ labeled formulas. In the best case, one might use $\mathsf{Ceiling}(\log_2 \mathsf{Max}(d, n-d))$ separating formulas, besides identity, where $d = \mathsf{Card}(\mathcal{D})$, to pairwise distinguish the n truth-values of $\mathcal{L}$; in the worst case, $n-1$ such connectives will be needed. The case in which, say, $B^{\theta\odot}_F([\varphi_i]_{i=1}^j)$ turns out to be empty originates in fact a new closure rule, and in that case the rule $B^{\theta\odot}_T([\varphi_i]_{i=1}^j)$ can thus be omitted; the case in which $B^{\theta\odot}_T([\varphi_i]_{i=1}^j)$ turns out to be empty is entirely symmetric.

Tableaux are built as usual, by applying the above rules, given an initial sequence of 2-signed formulas, and a branch is said to be *closed* if its closure is obtained by the application of any of the (C#) rules, including (C0). Branches that are not closed are said to be *open*. A tableau is said to be *closed* in case all of its branches are closed. By the construction of the above rules, it is easy to check the following soundness result with respect to the initially given truth-functional semantics:

Theorem 1. If a valuation satisfies some initial sequence of 2-signed formulas, then it satisfies all the formulas in some open branch of any tableau that originates from that set of formulas.

To enforce the requirement of analyticity for our tableaux the following strategy will be helpful. Notice that, from the point of view of analyticity, only a formula in the set $\Theta = \{\theta_r(\varphi) : 0 < r \leq s, \varphi \in \mathbb{S} \setminus \mathcal{A}\}$ is possibly 'problematic', as more than one rule may apply to (an appropriate labelling of) it. In the case of such a formula we will always apply a carefully chosen θ-rule, as follows. In general, given $\delta \in \Theta$, let $I_\delta = \{0 < r \leq s :$ there exists $\odot_r([\delta_i]_{i=1}^{j_r}) \in \mathbb{S}$ such that $\delta = \theta_r(\odot_r([\delta_i]_{i=1}^{j_r}))\}$. To ensure the termination of the tableau construction procedure, the choice of rule to be applied in each case will be guided by the following complexity measure $\ell : \mathbb{S} \to \mathbb{N}$, defined for the formulas of $\mathcal{L}$:

$$
\begin{array}{ll}
(\ell 0)\ \ell(p) = \ell(\theta_r(p)) = 0, & \text{where } p \in \mathcal{A};\\
(\ell 1)\ \ell(\delta) = 1 + \mathsf{Min}_{r \in I_\delta}\left(\sum_{i=1}^{j_r} \ell(\delta_i)\right), & \text{where } \delta \in \Theta;\\
(\ell 2)\ \ell(\odot([\varphi_i]_{i=1}^{j})) = 1 + \sum_{i=1}^{j} \ell(\varphi_i), & \text{otherwise.}
\end{array}
$$

Accordingly, no rule application should be allowed in a tableau for nodes of complexity 0; moreover, applications of θ-rules should always precede applications of non-θ-rules, as the former clearly contribute more than the latter to decreasing the overall complexity of the corresponding nodes. A particularly interesting situation arises in the case where the heads of more than one θ-rule are matched by the same node. In that case, the θ-rule to be applied may be chosen by heeding the minimality requirement in clause $(\ell 1)$ of the definition of the complexity measure. This choice is typically very simple. Indeed, consider the situation in which a formula δ can be obtained either as $\theta_i(\varphi_i)$ or as $\theta_j(\varphi_j)$. In that case, $\theta_i(p_i)$ might be thought of as a formula on the variable p_i, of which $\theta_i(\varphi_i)$ is a substitution instance, and similarly for $\theta_j(p_j)$ as a formula on the variable p_j. Now, by Robinson's unification algorithm, $\theta_i(p_i)$ and $\theta_j(p_j)$ will have a most general unifier, that is, either it is the case that $p_i = \sigma(p_j)$ or else that $p_j = \sigma(p_i)$, for an appropriate substitution σ. In the former case, where $p_i = \sigma(p_j)$, the θ-rule to be applied first is the one for θ_j, as $\theta_j(p_j) = \theta_i(\sigma(p_j))$; the latter case is entirely symmetric. As it can easily be seen, this prioritary application of the 'most concrete' rule will provide the greater decrease in complexity for a given node, and will help implementing the requirement of analyticity. The situation is a bit more complex in the remaining case, where the most general unifier of $\theta_i(p_i)$ and $\theta_j(p_j)$ asserts simultaneously that $\sigma(p_i) = \delta_i$ and $\sigma(p_j) = \delta_j$ where δ_i, δ_j are formulas in which the variables p_i, p_j, respectively, do not occur. In this (peculiar) case, and only for the formula $\theta_i(\delta_i) = \theta_j(\delta_j)$, we must directly check which of the two corresponding θ-rules matches the minimality requirement.

Say that a tableau branch is *exhausted* if it is closed, or if the appropriate θ-rules have been applied to every node whose formulas have non-null complexity and non-θ-rules have been applied to every other non-atomic node. Our main normalization result guarantees that:

Lemma 1. Exhausted tableaux always exist.

A nice thing about exhausted tableaux is that all counter-models for non-valid inferences can be built from them. Indeed, using the above lemma one may easily prove now the following completeness result:

Theorem 2. From every open branch of an exhausted tableau for a given initial sequence of 2-signed formulas one may extract a valuation satisfying those formulas.

As an immediate byproduct of the previous results, it follows that:

Corollary 1. For a given logic $\mathcal{L}$ with semantics Sem:

$\gamma_1, \ldots, \gamma_m \models_{\mathsf{Sem}} \alpha$ iff there is a closed tableau for $T{:}\gamma_1, \ldots, T{:}\gamma_m, F{:}\alpha$.

5 Future Work

There are a number of possible directions for further extension of our present results. Previous work on extraction of non-analytic tableau rules for sufficiently expressive finite-valued logics (cf. [6]) has been implemented (cf. [13]) for the extraction of appropriate axioms in the framework of `Isabelle`'s intuitionistic higher-order logic. A similar implementation might now be performed for the presently illustrated procedure, with the advantage that analyticity guarantees the existence of fully automated decision procedures, and these can be constructively assembled as tacticals in `Isabelle`'s meta-language. Moreover, the 2-signed 'normal form' that underlies the statement of our tableau rules in the (intuitionistic) meta-language may also be used as a basis for the study of classic-like automated reasoning mechanisms based on satisfiability checking or signed resolution (cf. [11,4,9]).

On what concerns the issue about 'sufficient expressiveness', a requisite for the application of our present procedure to a given finite-valued logic, it should be noted that the calculation of a convenient collection of separating formulas, whenever it exists, may also be performed automatically. Moreover, as it can be shown, for the logics that turn out *not* to be sufficiently expressive, a conservative extension of the original language may be devised by the addition of a convenient number of 0-ary connectives in order to make such logics amenable to our method. Such is the case, for instance, for Gödel-Dummett n-valued logics, with finite $n > 3$. The complexity of such procedures, not yet described nor implemented in detail, is still to be fully explored.

Finally, it will be interesting to investigate the amount in which the above procedures can be extended so as to apply to logics characterized by semantics that broaden the notion of truth-functionality, such as non-deterministic semantics (cf. [2]) and possible-translations semantics (cf. [12]). For one thing, as we shall show in future studies, the adequacy results that connect bivalent semantics to the corresponding tableau systems can in fact be generalized so as to cover many other logics characterized by what we call 'dyadic semantics' [6,7], including numerous interesting infinite-valued logics. Such generic results, in fact, also take into account logics whose non-truth-functional semantics is formulated using more than 2 'logical values'.

Acknowledgments. The first author was partially supported by FCT and EU FEDER via the KLog project PTDC/MAT/68723/2006 of SQIG-IT. The second author acknowledges partial support by SQIG-IT and CNPq. The authors are deeply indebted to their colleagues W. Carnielli and M. Coniglio for many fruitful discussions on topics related to this work, and also to D. Mendonça and three anonymous referees for their attentive reading and valuable suggestions of improvements on earlier drafts of the paper.

References

1. Aguzzoli, S., Ciabattoni, A., Di Nola, A.: Sequent calculi for finite-valued Łukasiewicz logics via Boolean decompositions. Journal of Logic and Computation 10(2), 213–222 (2000)
2. Avron, A., Lev, I.: Non-deterministic multiple-valued structures. Journal of Logic and Computation 15, 241–261 (2005)
3. Surma, S.J.: An algorithm for axiomatizing every finite logic. In: Rine, D.C. (ed.) Computer Science and Multiple-Valued Logics, Selected Papers from the IV International Symposium on Multiple-Valued Logics, 2nd edn., pp. 143–149. North-Holland, Amsterdam (1974) 2nd edn. (1984)
4. Baaz, M., Fermüller, C.G., Salzer, G.: Automated deduction for many-valued logics. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 1355–1402. Elsevier and MIT Press (2001)
5. Béziau, J.-Y.: Sequents and bivaluations. Logique et Analyse (N.S.) 44(176), 373–394 (2001)
6. Caleiro, C., Carnielli, W., Coniglio, M.E., Marcos, J.: Two's company: The humbug of many logical values. In: Béziau, J.-Y. (ed.) Logica Universalis, pp. 169–189. Birkhäuser Verlag, Basel (2005), http://wslc.math.ist.utl.pt/ftp/pub/CaleiroC/05-CCCM-dyadic.pdf
7. Caleiro, C., Carnielli, W.A., Coniglio, M.E., Marcos, J.: How many logical values are there? Dyadic semantics for many-valued logics. Draft (forthcoming) (2005)
8. Carnielli, W.A.: Systematization of the finite many-valued logics through the method of tableaux. The Journal of Symbolic Logic 52(2), 473–493 (1987)
9. Fermüller, C.G., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Handbook of Automated Reasoning, pp. 1791–1849. Elsevier, Amsterdam (2001)
10. Hähnle, R.: Tableaux for many-valued logics. In: D'Agostino, M., Gabbay, D., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods, pp. 529–580. Springer, Heidelberg (1999)
11. Hähnle, R.: Advanced many-valued logics. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 2, pp. 297–395. Kluwer, Dordrecht (2001)
12. Marcos, J.: Possible-translations semantics. In: Carnielli, W.A., Dionísio, F.M., Mateus, P. (eds.) Proceedings of the Workshop on Combination of Logics: Theory and applications (CombLog 2004), held in Lisbon, PT. 1049-001 Lisbon, PT, 2004. Departamento de Matemática, Instituto Superior Técnico. July 28–30, 2004, Lisbon, PT, July 28–30, 2004, pp. 119–128 (2004), http://wslc.math.ist.utl.pt/ftp/pub/MarcosJ/04-M-pts.pdf

13. Marcos, J., Mendonça, D.: Towards fully automated axiom extraction for finite-valued logics. In: Carnielli, W., Coniglio, M.E., D'Ottaviano, I.M.L. (eds.) The Many Sides of Logic, London. Studies in Logic. College Publications (2009), http://wslc.math.ist.utl.pt/ftp/pub/MarcosJ/08-MM-towards.pdf
14. Rosser, J.B., Turquette, A.R.: Many-Valued Logics. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam (1952)
15. Suszko, R.: The Fregean Axiom and Polish mathematical logic in the 1920s. Studia Logica 36, 373–380 (1977)
16. Wójcicki, R.: Theory of Logical Calculi. Kluwer, Dordrecht (1988)

Appendix

Illustration of a Tableau for Ł$_3$

The following example employs the simplified rules (Ł$_3$.#)*, together with the corresponding closure rules (Ł$_3$.C0) and (Ł$_3$.C1):

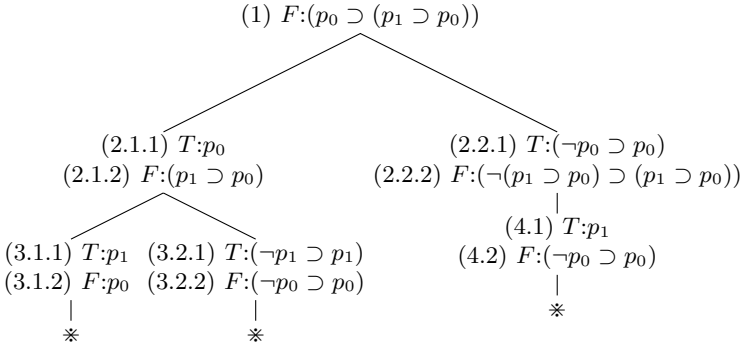


Fig. 1. A failed attempt to refute $p_0 \supset (p_1 \supset p_0)$

Proofs of the Main Results

Proof of Theorem 1. First, observe that if a valuation $\S : \mathbb{S} \to \mathcal{V}_n$ satisfies the head of a rule

$$X{:}\theta(\odot([\varphi_i]_{i=1}^{j})) \implies \| \, B_X^{\theta\odot}([\varphi_i]_{i=1}^{j})$$

then, by construction, one may conclude that $\S$ also satisfies $\&[\overline{v_i}^{\mathbb{S}}(\varphi_i)]_{i=1}^{j}$ for some sequence $[\overline{v_i}^{\mathbb{S}}(\varphi_i)]_{i=1}^{j}$ of labeled formulas such that $t(\widehat{\theta}(\widehat{\odot}([v_i]_{i=1}^{j}))) = X$, that is, $b_{\S}(\varphi_i) = v_i$, for $1 \leq i \leq j$.

Suppose now that a valuation $\S$ satisfies a given sequence of 2-signed formulas. Then, by the above observation, it is clear that $\S$ satisfies all the formulas in some branch of any tableau built from these initial formulas by applying rules as above. To see that any such branch must be open, just note that no closure rule may be applied. To that effect, the branch would have to contain labeled

formulas that match the head $F{:}\varphi$ & $T{:}\varphi$ of the classic-like closure rule (C0), or else the head $\&(\widetilde{c}^{\mathbb{S}}_R(\varphi))$ of a closure rule (C#), for some given minimal closuring partial binary print $\widetilde{c}_R$. It would follow then that any valuation that satisfies this branch would either have to assign two different values to the formula φ or have to associate to φ a closuring partial binary print.

Proof of Lemma 1. We first check that every rule applied according to the specified requirement of analiticity does indeed reduce the complexity.

- Consider $\varphi = \theta(\odot([\varphi_i]_{i=1}^j)) \in \Theta$ with $\ell(\varphi) = 1 + \ell(\varphi_1) + \ldots + \ell(\varphi_j)$, and the corresponding θ-rule:
$$X{:}\theta(\odot([\varphi_i]_{i=1}^j)) \implies \| \, B_X^{\theta\odot}([\varphi_i]_{i=1}^j).$$
Then, for each $1 \leq i \leq j$ and $0 \leq r \leq s$, we have that:
$\ell(\varphi) = 1 + \ell(\varphi_1) + \ldots + \ell(\varphi_j) > \ell(\varphi_i) \geq \ell(\theta_r(\varphi_i))$.
- Consider now $\varphi = \odot([\varphi_i]_{i=1}^j) \notin \Theta$, and the corresponding non-θ-rule:
$$X{:}\odot([\varphi_i]_{i=1}^j) \implies \| \, B_X^{\odot}([\varphi_i]_{i=1}^j).$$
Then, for each $1 \leq i \leq j$ and $0 \leq r \leq s$, we again have that:
$\ell(\varphi) = 1 + \ell(\varphi_1) + \ldots + \ell(\varphi_j) > \ell(\varphi_i) \geq \ell(\theta_r(\varphi_i))$.

In either case the complexity of every signed formula in the conclusion of the rule is lower than the complexity of the head signed formula. Hence, given an initial finite set with m many 2-signed formulas of complexity bounded by g, the tableau will be exhausted after the application of at most $m \times u^g$ rules, where u is the maximum number of formulas in the conclusion of a rule.

Proof of Theorem 2. Given an open branch of an exhausted tableau, consider any assignment $\rho : \mathcal{A} \to \mathcal{V}_n$ such that, for every $p \in \mathcal{A}$, the binary print $\overline{\rho(p)} = [X_i]_{i=0}^s$ of $\rho(p)$ agrees with the information available, that is:

- either $X_i{:}\theta_i(p)$ occurs in the branch, or
- neither $T{:}\theta_i(p)$ nor $F{:}\theta_i(p)$ occur in the branch.

Accordingly, if $X_i = T$ then $F{:}\theta_i(p)$ does not appear in the branch; *mutatis mutandis*, if $X_i = F$ then $T{:}\theta_i(p)$ does not appear in the branch. Note that since none of the closure rules can be applied to the branch, we have non-closuring information about every atom p, and such an assignment ρ can always be defined. To prove that the homomorphic extension $\S_\rho : \mathbb{S} \to \mathbb{V}$ of such assignment indeed satisfies all the formulas in the branch it is sufficient to prove that if $\S_\rho$ satisfies all the formulas in some branch of a tableau rule, then it also satisfies the head of the rule. To such purpose, we need only consider the generic case of a rule
$$X{:}\theta(\odot([\varphi_i]_{i=1}^j)) \implies \| \, B_X^{\theta\odot}([\varphi_i]_{i=1}^j).$$
Assume that $\S_\rho$ satisfies one of the disjunctions in the conclusion of the rule, that is, $\S_\rho$ satisfies some $B_X^{\theta\odot}([\varphi_i]_{i=1}^j) = \{\&[\overline{v_i}^{\mathbb{S}}(\varphi_i)]_{i=1}^j : t(\widehat{\theta}(\widehat{\odot}([v_i]_{i=1}^j))) = X\}$. Then, for each $1 \leq i \leq j$, $\S_\rho$ satisfies $\overline{v_i}^{\mathbb{S}}(\varphi_i)$ or, equivalently, $\S_\rho(\varphi_i) = v_i$. Therefore, it follows that $t(\S_\rho(\theta(\odot([\varphi_i]_{i=1}^j)))) = X$, and thus $\S_\rho$ satisfies the head of the rule $X{:}\theta(\odot([\varphi_i]_{i=1}^j))$.

A Duality for Algebras of Lattice-Valued Modal Logic

Yoshihiro Maruyama

Department of Humanistic Informatics, Graduate School of Letters,
Kyoto University, Japan
maruyama@i.h.kyoto-u.ac.jp

Abstract. In this paper, we consider some versions of Fitting's L-valued logic and L-valued modal logic for a finite distributive lattice L. Using the theory of natural dualities, we first obtain a natural duality for algebras of L-valued logic (i.e., L-**VL**-algebras), which extends Stone duality for Boolean algebras to the L-valued case. Then, based on this duality, we develop a Jónsson-Tarski-style duality for algebras of L-valued modal logic (i.e., L-**ML**-algebras), which extends Jónsson-Tarski duality for modal algebras to the L-valued case. By applying these dualities, we obtain compactness theorems for L-valued logic and for L-valued modal logic, and the classification of equivalence classes of categories of L-**VL**-algebras for finite distributive lattices L.

1 Introduction

Fitting [16] introduced L-valued logic and L-valued modal logics for a finite distributive lattice L, where all the elements of L are encoded as truth constants in the languages of Fitting's logics. L-valued modal logics are particularly important for the study of reasoning about knowledge and belief (see [18,20,24]). Fitting's logics have been studied from different perspectives. Proof-theoretic aspects are studied in the papers [16,17,19]. Model-theoretic aspects are studied in the papers [13,21,22]. However, there are few papers on algebraic aspects of them.

Although [13] mentions the difficulty of developing algebraic semantics for Fitting's logics, [23] provides algebraic semantics for Fitting's logics modified by replacing truth constants (except 0 and 1) with unary connectives T_a's for all $a \in L$, where $\mathrm{T}_a(x)$ intuitively means that the truth value of a proposition x is a (see Definition 1). For motivations of the modification, see [23, Section 1]. We remark that Fitting's L-valued logic modified in the above manner can be considered not only as a many-valued logic but also as a modal logic, which is discussed in Section 4. The class of L-**VL**-algebras introduced in [23] is the algebraic counterpart of Fitting's L-valued logic modified in the above manner. The class of L-**ML**-algebras introduced in [23] is the algebraic counterpart of Fitting's L-valued modal logic modified in the above manner.

In this paper, we develop topological dualities for L-**VL**-algebras and for L-**ML**-algebras, following Stone's maxim: "One must always topologize" (see [25]). However, why must one always topologize?

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 281–295, 2009.

From the viewpoint of mathematical logic, we give two reasons related to results of this paper: (1) In general, one of the advantages of a topological duality for algebras of a logic is that we can understand the geometric meanings of the logic and its properties. For example, by a topological duality for algebras of a logic, we often notice that the compactness of the logic can be represented by the compactness of the dual space of the Lindenbaum algebra of the logic, which holds true of dualities developed in this paper. By using the dualities in this paper, we can actually show compactness theorems for L-valued logic and for L-valued modal logic (see Theorem 2 and Theorem 7). (2) Another advantage is that a duality for algebras of a logic often leads us to a completeness result for the logic, which holds true of dualities developed in this paper, and therefore topological duality contributes to study of completeness, which is one of the most important properties of a logic.

From the algebraic viewpoint, topological duality makes it possible to solve purely algebraic problems more easily by topological methods than by algebraic methods. In this paper, we can actually obtain the classification of equivalence classes of categories of L-**VL**-algebras for finite distributive lattices L by considering the dual topological categories of those categories via a duality for L-**VL**-algebras (see Theorem 3).

From the viewpoint of computer science, topological dualities provide useful tools for studying the semantics of programming languages (for example, see [1,5,4]). In particular, there is an application of Jónsson-Tarski duality for modal algebras (see [3,9,12]) to program semantics in [5].

Finally, we remark that topological dualities play fundamental roles also in other areas of mathematics, such as algebraic geometry, functional analysis and non-commutative geometry (see [11,10,8]).

In the following two paragraphs, we briefly explain how to develop dualities for L-**VL**-algebras and for L-**ML**-algebras.

In order to develop a duality for L-**VL**-algebras, we use the theory of natural dualities (see [7]), which is a general theory of a duality for the quasi-variety generated by a finite algebra and therefore can be considered as a general theory of a duality for algebras of M-valued logic for a finite algebra M. Since a finite distributive lattice L endowed with T_a's for all $a \in L$ forms a semi-primal algebra (see Definition 7), it follows from results in [23] and the theory of natural dualities that a natural (strong) duality holds for L-**VL**-algebras (Theorem 1), which confirms the applicability of the theory of natural dualities. By letting L be the two-element Boolean algebra, we can recover Stone duality for Boolean algebras from the duality for L-**VL**-algebras.

Based on the natural duality for L-**VL**-algebras, we develop a Jónsson-Tarski-style duality for L-**ML**-algebras (Theorem 6), where "Jónsson-Tarski-style" means that, as in Jónsson-Tarski duality for modal algebras (see [3,9,12]), we equip the dual space of an L-**ML**-algebra with a certain relation (see Definition 16), by which we can define a modal operator $\Box$ on the double dual of an L-**ML**-algebra. The Jónsson-Tarski-style duality for L-**ML**-algebras is an L-valued version of Jónsson-Tarski duality for modal algebras. By letting L be the

two-element Boolean algebra, we can recover Jónsson-Tarski duality for modal algebras from the duality for L-**ML**-algebras. For a related result, we refer to a duality presented in [27]. Whereas the lattice of truth values is $\{0, 1/n, 2/n, ..., 1\}$ for $n \in \omega$ with $n \geq 1$ in [27], the lattice of truth values is an arbitrary (not necessarily totally ordered) finite distributive lattice in this paper. Hence, the duality developed in this paper seems to be more general than that in [27]. Note also that [27] considers an Łukasiewicz-style many-valued modal logic, while a Fitting-style many-valued modal logic is considered in this paper.

According to the ideas in [23, Section 4], we could develop different dualities for L-**VL**-algebras and L-**ML**-algebras. The relationships between the dualities developed in this paper and the dualities suggested in [23] are discussed in Section 4. Roughly speaking, the modified L-valued logic is considered as a many-valued logic in the dualities developed in this paper, while it is considered as a modal logic in the dualities suggested in [23] by replacing T_a's with U_a's (see Proposition 3), where U_a's are inter-definable with T_a's ([23, Proposition 7]) and can be considered as modalities ([23, Proposition 8]). The modified L-valued logic is "schizophrenic" in this sense.

2 A Natural Duality for L-VL-Algebras

In this paper, L denotes a finite distributive lattice with the greatest element 1 and the least element 0, where we assume $0 \neq 1$. Then L forms a finite Heyting algebra. For $a, b \in L$, let $a \to b$ denote the pseudo-complement of a relative to b.

Let **2** denote the two-element Boolean algebra.

Definition 1. *Endow L with unary operations $\mathrm{T}_a(\text{-})$'s for all $a \in L$ defined by*

$$\mathrm{T}_a(x) = \begin{cases} 1 & (\textit{if } x = a) \\ 0 & (\textit{if } x \neq a) \end{cases}$$

L-valued logic L-**VL** is defined as follows. The connectives of L-**VL** are $\wedge$, $\vee$, $\to$, 0, 1 and T_a for each $a \in L$, where every T_a is a unary connective, 0 and 1 are nullary connectives, and the others are binary connectives. **PV** denotes the set of propositional variables. The formulas of L-**VL** are recursively defined in the usual way. **Form** denotes the set of formulas of L-**VL**.

Definition 2. *v is an L-valuation on* **Form** *iff v is a function from* **Form** *to L and satisfies the following:*

1. $v(\mathrm{T}_a(x)) = \mathrm{T}_a(v(x))$;
2. $v(x \wedge y) = \inf(v(x), v(y))$;
3. $v(x \vee y) = \sup(v(x), v(y))$;
4. $v(x \to y) = v(x) \to v(y)$;
5. $v(a) = a$ *for* $a = 0, 1$.

$x \in$ **Form** *is called valid in* L-**VL** *iff* $v(x) = 1$ *for any L-valuation v on* **Form**.

Definition 3. *Let $X \subset$ **Form**. Then, X is satisfiable iff there is an L-valuation v on* **Form** *such that $v(x) = 1$ for any $x \in X$.*

A compactness theorem for L-**VL** is proved in Theorem 2 below via a topological duality for L-**VL**-algebras, which are defined in the following subsection.

2.1 L-VL-Algebras and Their Spectra

We introduce L-**VL**-algebras, which provide a sound and complete semantics for L-valued logic L-**VL** as is shown in [23].

Let $x \leq y$ denote $x \wedge y = x$ and $x \leftrightarrow y$ denote $(x \to y) \wedge (y \to x)$.

Definition 4 ([23]). *$(A, \wedge, \vee, \to, \mathrm{T}_a(a \in L), 0, 1)$ is an L-**VL**-algebra iff it satisfies the following for any $a, b \in L$ and any $x, y \in A$:*

1. *$(A, \wedge, \vee, \to, 0, 1)$ forms a Heyting algebra;*
2. *$\mathrm{T}_a(x) \wedge \mathrm{T}_b(y) \leq \mathrm{T}_{a \to b}(x \to y) \wedge \mathrm{T}_{a \wedge b}(x \wedge y) \wedge \mathrm{T}_{a \vee b}(x \vee y)$;*
   $\mathrm{T}_b(x) \leq \mathrm{T}_{\mathrm{T}_a(b)}(\mathrm{T}_a(x))$;
3. *$\mathrm{T}_0(0) = 1$; $\mathrm{T}_a(0) = 0$ for $a \neq 0$;*
   $\mathrm{T}_1(1) = 1$; $\mathrm{T}_a(1) = 0$ for $a \neq 1$;
4. *$\bigvee\{\mathrm{T}_a(x)\ ;\ a \in L\} = 1$;*
   $\mathrm{T}_a(x) \wedge \mathrm{T}_b(x) = 0$ for $a \neq b$;
   $\mathrm{T}_a(x) \vee (\mathrm{T}_a(x) \to 0) = 1$;
5. *$\mathrm{T}_1(\mathrm{T}_a(x)) = \mathrm{T}_a(x)$; $\mathrm{T}_0(\mathrm{T}_a(x)) = \mathrm{T}_a(x) \to 0$;*
   $\mathrm{T}_b(\mathrm{T}_a(x)) = 0$ for $b \neq 0, 1$;
6. *$\mathrm{T}_1(x) \leq x$; $\mathrm{T}_1(x \wedge y) = \mathrm{T}_1(x) \wedge \mathrm{T}_1(y)$;*
7. *$\bigwedge_{a \in L}(\mathrm{T}_a(x) \leftrightarrow \mathrm{T}_a(y)) \leq x \leftrightarrow y$.*

*A homomorphism of L-**VL**-algebras is defined as a function which preserves all the operations $(\wedge, \vee, \to, \mathrm{T}_a(a \in L), 0, 1)$.*

$\mathrm{T}_a(x) \wedge \mathrm{T}_b(y) \leq \mathrm{T}_{a \vee b}(x \vee y)$ intuitively means that if the truth value of x is a and the truth value of y is b then the truth value of $x \vee y$ is $a \vee b$. The other inequalities following from the item 2 above can be explained in similar ways.

$\bigvee\{\mathrm{T}_a(x)\ ;\ a \in L\} = 1$ in the item 4 above is called the L-valued excluded middle, since the **2**-valued excluded middle coincides with the ordinary excluded middle (see [23, Proposition 2]).

2-VL-algebras coincide with Boolean algebras (see [23, Proposition 2]).

Definition 5. *Let A be an L-**VL**-algebra. For a subalgebra M of L, $\mathrm{Spec}_M(A)$ denotes the set of all homomorphisms from A to M. We equip $\mathrm{Spec}_M(A)$ with the topology generated by $\langle x \rangle$ for $x \in A$, where $\langle x \rangle = \{v \in \mathrm{Spec}_M(A)\,;\, v(x) = 1\}$.*

Note that $\mathrm{Spec}_M(A)$ is a subspace of $\mathrm{Spec}_L(A)$ for any subalgebra M of L and that $\mathrm{Spec}_{\mathbf{2}}(A)$ consists of all homomorphisms from A to **2**.

Definition 6. *Let A be an L-**VL**-algebra. Define $\mathcal{B}(A) = \{x \in A\,;\, \mathrm{T}_1(x) = x\}$. An element of $\mathcal{B}(A)$ is called a Boolean element of A.*

For a homomorphism $f : A \to B$ between L-**VL**-algebras, if $x \in A$ is a Boolean element, then $f(x) \in B$ is also a Boolean element.

Proposition 1. *Let A be an L-**VL**-algebra. Then, $\mathcal{B}(A)$ is a Boolean algebra.*

Proof. This is easily verified via completeness (see [23, Theorem 4]). Note that the operations are well-defined on $\mathcal{B}(A)$.

Proposition 2. *For an L-**VL**-algebra A, define $t_1 : \mathrm{Spec}_L(A) \to \mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A))$ by $t_1(v) = \mathrm{T}_1 \circ v$ for $v \in \mathrm{Spec}_L(A)$. Then, t_1 is a homeomorphism.*

Proof. Note that $t_1(v) \in \mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A))$ for $v \in \mathrm{Spec}_L(A)$. We show that t_1 is injective. Let $v \neq u$ for $v, u \in \mathrm{Spec}_L(A)$. Then $v(x) \neq u(x)$ for some $x \in A$. Let $a = v(x)$. Then $v(\mathrm{T}_a(x)) = 1$ and $u(\mathrm{T}_a(x)) = 0$. Thus we have $t_1(v) \neq t_1(u)$.

We show that t_1 is surjective. Let $u \in \mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A))$. Define $v : A \to L$ by $v(x) = a \Leftrightarrow u(\mathrm{T}_a(x)) = 1$, where note that $\mathrm{T}_a(x)$ is a Boolean element for any $x \in A$ and that v is well-defined as a function. We claim that v is a homomorphism. We show only $v(x \to y) = v(x) \to v(y)$ for $x, y \in A$, since the other cases are verified in similar ways. Let $v(x) = a$ and $v(y) = b$. Then $u(\mathrm{T}_a(x)) = 1$ and $u(\mathrm{T}_b(y)) = 1$, whence $u(\mathrm{T}_a(x) \wedge \mathrm{T}_b(y)) = 1$. Thus, by 2 in Definition 4, we have $u(\mathrm{T}_{a \to b}(x \to y)) = 1$ and so $v(x \to y) = a \to b = v(x) \to v(y)$. To complete the proof of the surjectivity of t_1, it remains to show $t_1(v) = u$. It is clear that if $u(x) = 1$ for $x \in \mathcal{B}(A)$, then $(t_1(v))(x) = 1$. If $u(x) = 0$ for $x \in \mathcal{B}(A)$, then $u(\mathrm{T}_0(\mathrm{T}_1(x))) = u(\mathrm{T}_1(x) \to 0) = u(x \to 0) = 1$ by 5 in Definition 4, whence $v(\mathrm{T}_1(x)) = 0$ and so $(t_1(v))(x) = 0$. Thus we have $t_1(v) = u$.

It is straightforward to verify the remaining part of the proof.

Let M be a subalgebra of L. Since M is a finite distributive lattice, it follows from the above proposition that $\mathrm{Spec}_M(A)$ is homeomorphic to $\mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A))$.

We define unary operations U_a as in the next proposition (for the proof, see [23]). $\mathrm{U}_a(x)$ intuitively means: The truth value of x is more than or equal to a. U_a's are used in the definition of L-**ML**-algebras (see Definition 15).

Proposition 3. *For $a \in L$, define $\mathrm{U}_a(x) = \bigvee\{\mathrm{T}_b(x) \,;\, a \leq b\}$. For $x \in L$, we have $\mathrm{U}_a(x) = 1$ for $a \leq x$ and $\mathrm{U}_a(x) = 0$ for $a \not\leq x$.*

Note that U_a is order-preserving and that $\mathrm{U}_a(x \wedge y) = \mathrm{U}_a(x) \wedge \mathrm{U}_a(y)$.

In fact, U_a's are inter-definable with T_a's (see [23, Proposition 7]).

2.2 A Natural Duality for L-VL-Algebras

As in universal algebra, we mean by an algebra a set A endowed with a collection of operations on A (see [6,7]). For an algebra A, $\mathbb{ISP}(A)$ denotes the class of all isomorphic copies of subalgebras of direct powers of A (see [6,7]).

Definition 7 ([7]). *Let A be a finite algebra. Define the ternary discriminator function $t : A^3 \to A$ on A as follows:*

$$t(x, y, z) = \begin{cases} x & \text{if } x \neq y \\ z & \text{if } x = y. \end{cases}$$

A is called a semi-primal algebra iff t is a term function of A and there is no isomorphism between non-empty subalgebras of A other than the identity maps.

Lemma 1. $(L, \wedge, \vee, \to, \mathrm{T}_a(a \in L), 0, 1)$ *forms a semi-primal algebra.*

Proof. Define $t : L^3 \to L$ by

$$t(x, y, z) = ((\mathrm{T}_1(x \leftrightarrow y) \to 0) \to x) \wedge (\mathrm{T}_1(x \leftrightarrow y) \to z).$$

It is easily verified that t is actually the ternary discriminator function on L.

Suppose for contradiction that there exists an isomorphism $f : L_1 \to L_2$ such that both L_1 and L_2 are non-empty subalgebras of L and that $f(a) \neq a$ for some $a \in L_1$. Then, we have $1 = f(\mathrm{T}_a(a)) = \mathrm{T}_a(f(a)) = 0$, which is a contradiction.

Definition 8. L-VA *denotes the category of* L-**VL**-*algebras and homomorphisms of* L-**VL**-*algebras.*

A Boolean space is defined as a zero-dimensional compact Hausdorff space.

Definition 9. $\mathrm{SubAlg}(L)$ *denotes the set of all subalgebras of* L*. For a Boolean space* S*,* $\mathrm{SubSp}(S)$ *denotes the set of all closed subspaces of* S*.*

Note that a closed subspace of a Boolean space is also a Boolean space.

Definition 10. *We define a category* L-BS *as follows.*

An object in L-BS *is a tuple* (S, α) *such that* S *is a Boolean space and that a function* $\alpha : \mathrm{SubAlg}(L) \to \mathrm{SubSp}(S)$ *satisfies the following:*

1. $S = \alpha(L)$*;*
2. *if* L_1 *is a subalgebra of* L_2*, then* $\alpha(L_1) \subset \alpha(L_2)$*;*
3. *if* $L_3 = L_1 \cap L_2$*, then* $\alpha(L_3) = \alpha(L_1) \cap \alpha(L_2)$*.*

An arrow $f : (S, \alpha) \to (T, \beta)$ *in* L-BS *is a continuous map* $f : S \to T$ *which satisfies the condition that, for any* $M \in \mathrm{SubAlg}(L)$*, if* $x \in \alpha(M)$ *then* $f(x) \in \beta(M)$*. We call a map satisfying the condition a subspace-preserving map.*

Definition 11. *We define a contravariant functor* $\mathrm{Spec} : L\text{-VA} \to L\text{-BS}$*.*

For an object A *in* L-VA*, define* $\mathrm{Spec}(A) = (\mathrm{Spec}_L(A), \alpha_A)$*, where* α_A *is defined by* $\alpha_A(M) = \mathrm{Spec}_M(A)$ *for* $M \in \mathrm{SubAlg}(L)$*.*

For an arrow $f : A \to B$ *in* L-VA*,* $\mathrm{Spec}(f) : \mathrm{Spec}(B) \to \mathrm{Spec}(A)$ *is defined by* $\mathrm{Spec}(f)(v) = v \circ f$ *for* $v \in \mathrm{Spec}_L(B)$*.*

We equip L and its subalgebras with the discrete topologies. Define $\alpha_L : \mathrm{SubAlg}(L) \to \mathrm{SubSp}(L)$ by $\alpha_L(M) = M$ for $M \in \mathrm{SubAlg}(L)$. Then, (L, α_L) is an object in L-BS.

Definition 12. *We define a contravariant functor* $\mathrm{Cont} : L\text{-BS} \to L\text{-VA}$*.*

For an object (S, α) *in* L-BS*,* $\mathrm{Cont}(S, \alpha)$ *is defined as the set of all subspace-preserving continuous maps from* (S, α) *to* (L, α_L) *endowed with the operations* $(\wedge, \vee, \to, \mathrm{T}_a(a \in L), 0, 1)$ *defined pointwise: For* $f, g \in \mathrm{Cont}(S, \alpha)$*, define* $(f @ g)(x) = f(x) @ g(x)$ *for* $@ = \wedge, \vee, \to$*. Define* $(\mathrm{T}_a(f))(x) = \mathrm{T}_a(f(x))$*. Finally* 0 *(resp.* 1*) is defined as the constant function whose value is always* 0 *(resp.* 1*).*

For an arrow $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ *in* L-BS*,* $\mathrm{Cont}(f) : \mathrm{Cont}(S_2, \alpha_2) \to \mathrm{Cont}(S_1, \alpha_1)$ *is defined by* $\mathrm{Cont}(f)(g) = g \circ f$ *for* $g \in \mathrm{Cont}(S_2, \alpha_2)$*.*

The following is a natural duality for L-**VL**-algebras.

Theorem 1. *The category* L-VA *is dually equivalent to the category* L-BS *via the functors* Spec(-) *and* Cont(-).

Proof. Let Id_1 denote the identity functor on L-VA and Id_2 denote the identity functor on L-BS. It is sufficient to show that there are two natural isomorphisms $\epsilon : \mathrm{Id}_1 \to \mathrm{Cont} \circ \mathrm{Spec}$ and $\eta : \mathrm{Id}_2 \to \mathrm{Spec} \circ \mathrm{Cont}$.

For an L-**VL**-algebra A, define $\epsilon_A : A \to \mathrm{Cont} \circ \mathrm{Spec}(A)$ by $\epsilon_A(x)(v) = v(x)$, where $x \in A$ and $v \in \mathrm{Spec}_L(A)$.

For an object (S, α) in L-BS, define $\eta_{(S,\alpha)} : (S, \alpha) \to \mathrm{Spec} \circ \mathrm{Cont}(S, \alpha)$ by $\eta_{(S,\alpha)}(x)(f) = f(x)$, where $x \in S$ and $f \in \mathrm{Cont}(S, \alpha)$.

Since L forms a semi-primal algebra by Lemma 1 and since the class of L-**VL**-algebras coincides with $\mathbb{ISP}(L)$ by [23, Theorem 3], it follows from [7, Theorem 3.3.14] that η and ϵ are natural isomorphisms.

By Theorem 1, we can show a compactness theorem for L-**VL**.

Theorem 2. *Let* $X \subset$ **Form**. *Assume that any finite subset of* X *is satisfiable. Then,* X *is satisfiable.*

Proof. Let A be the Lindenbaum algebra of L-**VL**. We may consider $X \subset A$. Then $\{\langle x \rangle \ ; \ x \in X\}$ consists of clopen subsets of $\mathrm{Spec}_L(A)$. By assumption, $\{\langle x \rangle \ ; \ x \in X\}$ has finite intersection property, since an L-valuation on **Form** can be seen as a homomorphism from A to L. Since $\mathrm{Spec}_L(A)$ is compact by Theorem 1, there is $v \in \bigcap\{\langle x \rangle \ ; \ x \in X\}$. Hence, X is satisfiable.

By Theorem 1, we can also classify the equivalence classes of the categories L-VA's for finite distributive lattices L as follows (for definitions from category theory, we refer to [2]).

Theorem 3. *Let* L_1 *and* L_2 *be finite distributive lattices. Then, the following are equivalent:*

1. L_1-VA *and* L_2-VA *are categorically equivalent;*
2. $(\mathrm{SubAlg}(L_1), \subset)$ *and* $(\mathrm{SubAlg}(L_2), \subset)$ *are order isomorphic.*

Proof. We show that (2) implies (1). Assume (2). Then, L_1-BS and L_2-BS are categorically equivalent by the definition of L-BS. Thus, by Theorem 1, L_1-VA and L_2-VA are categorically equivalent.

We show that (1) implies (2). Assume (1). Then, by Theorem 1, L_1-BS and L_2-BS are categorically equivalent. For $i = 1, 2$, equip $\{i\}$ with the discrete topology and define $\alpha_i : \mathrm{SubAlg}(L_i) \to \mathrm{SubSp}(\{i\})$ by $\alpha_i(M) = \{i\}$ for any $M \in \mathrm{SubAlg}(L_i)$. Then, $(\{i\}, \alpha_i)$ is a terminal object of L_i-BS for $i = 1, 2$. Let $\mathrm{SubOb}(\{i\}, \alpha_i)$ denote the category of non-empty subobjects of $(\{i\}, \alpha_i)$ in L_i-BS (for the definition of the category of subobjects, see [2, Definition 5.1]). Since L_1-BS and L_2-BS are categorically equivalent, $\mathrm{SubOb}(\{1\}, \alpha_1)$ and $\mathrm{SubOb}(\{2\}, \alpha_2)$ are categorically equivalent. We consider $\mathrm{SubOb}(\{1\}, \alpha_1)$ and

$\mathrm{SubOb}(\{2\}, \alpha_2)$ as partially ordered sets in the usual way. Then, $\mathrm{SubOb}(\{1\}, \alpha_1)$ and $\mathrm{SubOb}(\{2\}, \alpha_2)$ are order isomorphic.

To complete the proof, it suffices to show that $\mathrm{SubOb}(\{i\}, \alpha_i)^{\mathrm{op}}$ is order isomorphic to $(\mathrm{SubAlg}(L_i), \subset)$ for $i = 1, 2$. For $(X, \alpha_X) \in \mathrm{SubOb}(\{i\}, \alpha_i)$, define

$$h_i(X, \alpha_X) = \bigcap \{M \in \mathrm{SubAlg}(L_i) \,;\, \alpha_X(M) = \{i\}\},$$

where we may assume $X = \{i\}$. Then we claim that h_i is an order isomorphism from $\mathrm{SubOb}(\{i\}, \alpha_i)^{\mathrm{op}}$ to $(\mathrm{SubAlg}(L_i), \subset)$. Since $\{M \,;\, \alpha_X(M) = \{i\}\}$ is upward closed with respect to $\subset$ by the item 2 of Definition 10 and since $\alpha_X(h_i(X, \alpha_X)) = \{i\}$ by the item 3 of Definition 10, h_i is injective. It is straightforward to verify the remaining part of the claim.

3 A Jónsson-Tarski-Style Duality for *L*-ML-Algebras

L-valued modal logic L-**ML** is defined by L-valued Kripke semantics as follows. The connectives of L-**ML** are a unary connective $\Box$ and the connectives of L-**VL**. Then, $\mathbf{Form}_\Box$ denotes the set of formulas of L-**ML**.

Definition 13. *Let (M, R) be a Kripke frame, i.e., R is a relation on a set M. Then, e is a Kripke L-valuation on (M, R) iff e is a function from $M \times \mathbf{Form}_\Box$ to L and satisfies the following for each $w \in M$ and $x \in \mathbf{Form}_\Box$:*

1. $e(w, \Box x) = \bigwedge \{e(w', x) \,;\, wRw'\}$;
2. $e(w, \mathrm{T}_a(x)) = \mathrm{T}_a(e(w, x))$;
3. $e(w, x @ y) = e(w, x) @ e(w, y)$ *for* $@ = \wedge, \vee, \rightarrow$;
4. $e(w, a) = a$ *for* $a = 0, 1$.

*We call (M, R, e) an L-valued Kripke model. $x \in \mathbf{Form}_\Box$ is said to be valid in L-**ML** iff $e(w, x) = 1$ for any L-valued Kripke model (M, R, e) and any $w \in M$.*

Definition 14. *Let $X \subset \mathbf{Form}_\Box$. X is Kripke-satisfiable iff there are an L-valued Kripke model (M, R, e) and $w \in M$ such that $e(w, x) = 1$ for any $x \in X$.*

A compactness theorem for L-**ML** is shown in Theorem 7 below.

3.1 *L*-ML-Algebras and Their Relational Spectra

We introduce L-**ML**-algebras, which provide a sound and complete semantics for L-valued modal logic L-**ML** as is shown in [23]. Recall that $\mathrm{U}_a(x)$ is the abbreviation of $\bigvee \{\mathrm{T}_b(x) \,;\, a \leq b\}$.

Definition 15 ([23]). *$(A, \wedge, \vee, \rightarrow, \mathrm{T}_a(a \in L), \Box, 0, 1)$ is an L-**ML**-algebra iff it satisfies the following:*

1. *$(A, \wedge, \vee, \rightarrow, \mathrm{T}_a(a \in L), 0, 1)$ forms an L-**VL**-algebra;*
2. *$\Box(x \wedge y) = \Box x \wedge \Box y$ and $\Box 1 = 1$;*
3. *$\Box \mathrm{U}_a(x) = \mathrm{U}_a(\Box x)$ for all $a \in L$.*

*A homomorphism of L-**ML**-algebras is defined as a homomorphism of L-**VL**-algebras which additionally preserves the operation $\Box$.*

Let A be an L-**ML**-algebra. Since $\mathrm{T}_1(x) = \mathrm{U}_1(x)$ for any $x \in A$, we have $\Box\mathrm{T}_1(x) = \mathrm{T}_1(\Box x)$ by the item 3 in the above definition.

Note that **2-ML**-algebras coincide with modal algebras.

Definition 16. *Let A be an L-**ML**-algebra. Define a relation $R_\Box$ on $\mathrm{Spec}_L(A)$ by*

$$vR_\Box u \Leftrightarrow \forall a \in L\, \forall x \in A\, (v(\Box x) \geq a \text{ implies } u(x) \geq a).$$

Define $e : \mathrm{Spec}_L(A) \times A \to L$ by $e(v, x) = v(x)$ for $v \in \mathrm{Spec}_L(A)$ and $x \in A$. Then, $(\mathrm{Spec}_L(A), R_\Box, e)$ is called the L-valued canonical model of A.

Proposition 4. *Let A be an L-**ML**-algebra. Then, the L-valued canonical model $(\mathrm{Spec}_L(A), R_\Box, e)$ of A is actually an L-valued Kripke model. In particular, $e(v, \Box x) = v(\Box x) = \bigwedge\{u(x)\,;\, vR_\Box u\}$ for $x \in A$ and $v \in \mathrm{Spec}_L(A)$.*

Proof. See [23, Proposition 10]. Note that, for an L-**VL**-algebra A, there is a natural bijection between the homomorphisms of L-**VL**-algebras from A to L and the "prime L-filters" of A (see [23, Proposition 5]). Prime L-filter gives an internal description of homomorphism into L.

Proposition 5. *Let A be an L-**ML**-algebra. Then $\mathcal{B}(A)$ is a modal algebra.*

Proof. By Proposition 1, it suffices to show that $\mathcal{B}(A)$ is closed under $\Box$. For $x \in \mathcal{B}(A)$, we have $\mathrm{T}_1(\Box x) = \Box\mathrm{T}_1(x) = \Box x$.

Proposition 6. *Let A be an L-**ML**-algebra and $v, u \in \mathrm{Spec}_L(A)$. Then the following holds: $vR_\Box u$ iff $t_1(v)R_\Box t_1(u)$, where $R_\Box$ in the right-hand side is defined only on $\mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A))$ (for the definition of t_1, see Proposition 2).*

Proof. From $\Box\mathrm{T}_1(x) = \mathrm{T}_1(\Box x)$, it follows that $vR_\Box u$ implies $t_1(v)R_\Box t_1(u)$. We show the converse. Assume $t_1(v)R_\Box t_1(u)$. In order to show $vR_\Box u$, it suffices to show that, for any $a \in L$ and any $x \in A$, $v(\Box\mathrm{U}_a(x)) \geq 1$ implies $u(\mathrm{U}_a(x)) \geq 1$, which follows from the assumption, since we have $\mathrm{U}_a(x) \in \mathcal{B}(A)$ and $\mathrm{T}_1(\mathrm{U}_a(x)) = \mathrm{U}_a(x)$.

By Proposition 2 and Proposition 6, we notice that t_1 is an "isomorphism" from $(\mathrm{Spec}_L(A), R_\Box)$ to $(\mathrm{Spec}_{\mathbf{2}}(\mathcal{B}(A)), R_\Box)$ for an L-**ML**-algebra A.

3.2 A Jónsson-Tarski-Style Duality for L-ML-Algebras

Definition 17. *L-*MA *denotes the category of L-**ML**-algebras and homomorphisms of L-**ML**-algebras.*

Definition 18. *Let (S, R) be a Kripke frame and f a function from S to L. Define $\Box_R f : S \to L$ by $(\Box_R f)(x) = \bigwedge\{f(y)\,;\, xRy\}$.*

For a Kripke frame (S, R) and $X \subset S$, let $R^{-1}[X] = \{y \in S\,;\, \exists x \in X\ yRx\}$. For $x \in S$, let $R[x] = \{y \in S\,;\, xRy\}$.

Definition 19. *We define a category* L-RS *as follows.*

An object in L-RS *is a triple* (S, α, R) *such that* (S, α) *is an object in* L-BS *and a relation* R *on* S *satisfies the following conditions:*

1. *if* $\forall f \in \mathrm{Cont}(S, \alpha)((\Box_R f)(x) = 1 \Rightarrow f(y) = 1)$ *then* xRy*;*
2. *if* $X \subset S$ *is clopen in* S*, then* $R^{-1}[X]$ *is clopen in* S*;*
3. *for any* $M \in \mathrm{SubAlg}(L)$*, if* $x \in \alpha(M)$ *then* $R[x] \subset \alpha(M)$*.*

An arrow $f : (S_1, \alpha_1, R_1) \to (S_2, \alpha_2, R_2)$ *in* L-RS *is an arrow* $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ *in* L-BS *satisfying the following conditions:*

1. *if* xR_1y *then* $f(x)R_2f(y)$*;*
2. *if* $f(x_1)R_2x_2$ *then there is* $y_1 \in S_1$ *such that* $x_1R_1y_1$ *and* $f(y_1) = x_2$*.*

The item 1 in the object part of Definition 19 is an L-valued version of the tightness condition of descriptive general frames in classical modal logic (for the definition of the tightness condition, see [9]).

Definition 20. *We define a contravariant functor* $\mathrm{RSpec} : L\text{-MA} \to L\text{-RS}$*. For an object* A *in* L-MA*, define* $\mathrm{RSpec}(A) = (\mathrm{Spec}_L(A), \alpha_A, R_\Box)$*. For an arrow* $f : A \to B$ *in* L-MA*, define* $\mathrm{RSpec}(f)$ *by* $\mathrm{RSpec}(f)(v) = v \circ f$ *for* $v \in \mathrm{Spec}_L(B)$*.*

The well-definedness of RSpec is shown by the following two lemmas.

Lemma 2. *Let* A *be an* L**-ML***-algebra. Then,* $\mathrm{RSpec}(A)$ *is an object in* L-RS*.*

Proof. First, we show that $\mathrm{RSpec}(A)$ satisfies the condition 1 in the object part of Definition 19. We show the contrapositive. Assume $(v_1, v_2) \notin R_\Box$ for $v_1, v_2 \in \mathrm{Spec}_L(A)$. Then there are $a \in L$ and $x \in A$ such that $v_1(\Box x) \geq a$ and $v_2(x) \not\geq a$, whence we have $v_1(\mathrm{U}_a(\Box x)) = 1$ and $v_2(\mathrm{U}_a(x)) \neq 1$. Define $f : \mathrm{Spec}_L(A) \to L$ by $f(v) = v(\mathrm{U}_a(x))$. Then, by Proposition 4 and $\Box \mathrm{U}_a(x) = \mathrm{U}_a(\Box x)$, we have

$$(\Box_R f)(v_1) = \bigwedge\{f(v)\,;\, v_1R_\Box v\} = \bigwedge\{v(\mathrm{U}_a(x))\,;\, v_1R_\Box v\} = v_1(\Box \mathrm{U}_a(x)) = 1.$$

By the definition of f, $f(v_2) \neq 1$. It is easy to verify that f is continuous.

Second, we show that $\mathrm{RSpec}(A)$ satisfies the condition 2. It is enough to show that $R_\Box^{-1}(\langle x \rangle)$ is clopen in S for any $x \in A$. We claim that

$$R_\Box^{-1}(\langle x \rangle) = \langle \neg\Box\neg \mathrm{T}_1(x) \rangle,$$

where $\neg x$ is the abbreviation of $x \to 0$. Note that the right-hand side is clopen. Assume $v \in \langle \neg\Box\neg \mathrm{T}_1(x) \rangle$. Then, $v(\neg\Box\neg \mathrm{T}_1(x)) = 1$ and so $v(\Box\neg \mathrm{T}_1(x)) = 0$. By Proposition 4, we have $0 = v(\Box\neg \mathrm{T}_1(x)) = \bigwedge\{u(\neg \mathrm{T}_1(x))\,;\, vR_\Box u\}$. Since $u(\neg \mathrm{T}_1(x))$ is either 0 or 1, there exists $u \in \mathrm{Spec}_L(A)$ with $vR_\Box u$ such that $u(\neg \mathrm{T}_1(x)) = 0$, i.e., $u(x) = 1$. Therefore we have $v \in R_\Box^{-1}(\langle x \rangle)$. The converse is proved in a similar way.

Third, we show that $\mathrm{RSpec}(A)$ satisfies the condition 3. Suppose for contradiction that $u \in \mathrm{Spec}_M(A)$ and $R[u] \setminus \mathrm{Spec}_M(A) \neq \emptyset$ for some $M \in \mathrm{SubAlg}(L)$. Then there is $v \in R[u] \setminus \mathrm{Spec}_M(A)$ and so there is $x_0 \in A$ with $v(x_0) \notin M$. Define $a = v(x_0)$. Then we have: For $w \in \mathrm{Spec}_L(A)$,

$$w(\mathrm{T}_a(x_0) \to x_0) = \begin{cases} 1 & \text{if } w(x_0) \neq a \\ a & \text{if } w(x_0) = a. \end{cases}$$

Therefore, it follows from Proposition 4 and uRv that

$$u(\Box(\mathrm{T}_a(x_0) \to x_0)) = \bigwedge\{w(\mathrm{T}_a(x_0) \to x_0)\,;\, uRw\} = a = v(x_0).$$

This contradicts $u \in \mathrm{Spec}_M(A)$ by $v(x_0) \notin M$, which completes the proof.

Lemma 3. *For L-**ML**-algebras A_1 and A_2, let $f : A_1 \to A_2$ be a homomorphism of L-**ML**-algebras. Then,* $\mathrm{RSpec}(f)$ *is an arrow in L-RS.*

Proof. By Theorem 1, $\mathrm{RSpec}(f)$ is an arrow in L-BS. Define $f_* : \mathcal{B}(A_1) \to \mathcal{B}(A_2)$ by $f_*(x) = f(x)$ for $x \in \mathcal{B}(A_1)$. By Proposition 5, f_* is a homomorphism between **2-ML**-algebras (i.e., modal algebras). Consider $\mathrm{RSpec}(f_*) : \mathrm{RSpec}(\mathcal{B}(A_2)) \to \mathrm{RSpec}(\mathcal{B}(A_1))$. It follows from Proposition 2, Proposition 6 and Jónsson-Tarski duality for modal algebras (see [12,3]) that $\mathrm{RSpec}(f_*)$ is an arrow in **2**-RS. We also have $\mathrm{RSpec}(f) = \mathrm{RSpec}(f_*)$ on $\mathrm{RSpec}(\mathcal{B}(A_2))$. By using these facts and Proposition 6, it is verified that $\mathrm{RSpec}(f)$ is an arrow in L-RS.

Definition 21. *We define a contravariant functor* $\mathrm{MCont} : L\text{-RS} \to L\text{-MA}$. *For an object (S, α, R) in L-RS, define* $\mathrm{MCont}(S, \alpha, R) = (\mathrm{Cont}(S, \alpha), \Box_R)$. *For an arrow $f : (S_1, \alpha_1, R_1) \to (S_2, \alpha_2, R_2)$ in L-RS, define* $\mathrm{MCont}(f)$ *by* $\mathrm{MCont}(f)(g) = g \circ f$ *for $g \in \mathrm{Cont}(S_2, \alpha_2)$.*

The well-definedness of MCont is shown by the following two lemmas.

Lemma 4. *Let (S, α, R) be an object in L-RS. Then,* $\mathrm{MCont}(S, \alpha, R)$ *is an L-**ML**-algebra.*

Proof. We first show that if $f \in \mathrm{MCont}(S, \alpha, R)$ then $\Box_R f \in \mathrm{MCont}(S, \alpha, R)$. Let $f \in \mathrm{MCont}(S, \alpha, R)$. Now we have the following: For $a \in L$,

$$(\Box_R f)^{-1}(a) = R^{-1}[(\mathrm{T}_a(f))^{-1}(1)] \cap (S \setminus R^{-1}[(\mathrm{U}_a(f))^{-1}(0)]),$$

where note: $x \in R^{-1}[(\mathrm{T}_a(f))^{-1}(1)]$ means that there is $y \in S$ such that xRy and $f(y) = a$; $x \in S \setminus R^{-1}[(\mathrm{U}_a(f))^{-1}(0)]$ means that there is no $y \in S$ such that xRy and $f(y) \not\geq a$. Since $R^{-1}[(\mathrm{T}_a(f))^{-1}(1)] \cap (S \setminus R^{-1}[(\mathrm{U}_a(f))^{-1}(0)])$ is clopen in S, $\Box_R f$ is a continuous map from S to L. It follows from the condition 3 in Definition 19 that $\Box_R f$ is subspace-preserving. Thus $\Box_R f \in \mathrm{MCont}(S, \alpha, R)$.

Next we show that $\mathrm{U}_a(\Box f) = \Box \mathrm{U}_a(f)$. It suffices to show that

$$\mathrm{U}_a(\bigwedge\{f(y)\,;\, xRy\}) = \bigwedge\{\mathrm{U}_a(f(y))\,;\, xRy\},$$

which is easily verified. The remaining part of the proof is also easy to check.

Lemma 5. *Let* $f : (S_1, \alpha_1, R_1) \to (S_2, \alpha_2, R_2)$ *be an arrow in* L-RS. *Then,* $\mathrm{MCont}(f)$ *is a homomorphism of* L-**ML**-*algebras.*

Proof. By Theorem 1, $\mathrm{MCont}(f)$ is an arrow in L-VA. It remains to show that $\mathrm{MCont}(f)(\Box g_2) = \Box(\mathrm{MCont}(f)(g_2))$ for $g_2 \in \mathrm{Cont}(S_2, \alpha_2)$. Let $x_1 \in S_1$. Then,

$$(\mathrm{MCont}(f)(\Box g_2))(x_1) = \Box g_2 \circ f(x_1) = \bigwedge\{g_2(y_2)\,;\, f(x_1)R_2y_2\}.$$

Let a denote the rightmost side of the above equation. We also have

$$(\Box(\mathrm{MCont}(f)(g_2)))(x_1) = (\Box(g_2 \circ f))(x_1) = \bigwedge\{g_2(f(y_1))\,;\, x_1R_1y_1\}.$$

Let b denote the rightmost side of the above equation. Since $x_1R_1y_1$ implies $f(x_1)R_1f(y_1)$, we have $a \leq b$. Since f satisfies the condition 2 in the arrow part of Definition 19, we have $a \geq b$. Hence $a = b$, which completes the proof.

Theorem 4. *Let* A *be an* L-**ML**-*algebra. Then,* A *is isomorphic to* $\mathrm{MCont} \circ \mathrm{RSpec}(A)$ *in the category* L-MA.

Proof. Define $\epsilon'_A : A \to \mathrm{MCont} \circ \mathrm{RSpec}(A)$ by $\epsilon'_A(x)(v) = v(x)$ for $x \in A$ and $v \in \mathrm{Spec}_L(A)$. Note that ϵ'_A is almost the same as ϵ_A in the proof of Theorem 1. By Theorem 1, ϵ'_A is an isomorphism of L-**VL**-algebras.

Therefore, it remains to show that ϵ'_A preserves $\Box$, i.e., $\epsilon'_A(\Box x) = \Box_{R_\Box}\epsilon'_A(x)$ for $x \in A$. For $v \in \mathrm{RSpec}(A)$, we have the following:

$$\begin{aligned}(\Box_{R_\Box}\epsilon'_A(x))(v) &= \bigwedge\{\epsilon'_A(x)(u)\,;\, vR_\Box u\}\\ &= \bigwedge\{u(x)\,;\, vR_\Box u\}\\ &= v(\Box x) \quad \text{(by Proposition 4)}\\ &= \epsilon'_A(\Box x)(v).\end{aligned}$$

This completes the proof.

Theorem 5. *Let* (S, α, R) *be an object in* L-RS. *Then,* (S, α, R) *is isomorphic to* $\mathrm{RSpec} \circ \mathrm{MCont}(S, \alpha, R)$ *in the category* L-RS.

Proof. Define $\eta'_{(S,\alpha,R)} : (S, \alpha, R) \to \mathrm{RSpec} \circ \mathrm{MCont}(S, \alpha, R)$ by $\eta'_{(S,\alpha,R)}(x)(f) = f(x)$ for $x \in S$ and $f \in \mathrm{Cont}(S, \alpha)$. Note that $\eta'_{(S,\alpha,R)}$ is almost the same as $\eta_{(S,\alpha)}$ in the proof of Theorem 1. By Theorem 1, $\eta'_{(S,\alpha,R)}$ is an isomorphism in the category L-BS. In the below, we denote $\eta'_{(S,\alpha,R)}$ by η'_S.

We show: For any $x, y \in S$, xRy iff $\eta'_S(x)R_{\Box_R}\eta'_S(y)$. Note that the right-hand side holds iff the following holds:

$$\forall a \in L\, \forall f \in \mathrm{Cont}(S, \alpha)\ (\eta'_S(x)(\Box_R f) \geq a \text{ implies } \eta'_S(y)(f) \geq a).$$

Assume xRy. Let $a \in L$ and $f \in \mathrm{Cont}(S, \alpha)$ with $\eta'_S(x)(\Box_R f) \geq a$. Since

$$a \leq \eta'_S(x)(\Box_R f) = (\Box_R f)(x) = \bigwedge\{f(z)\,;\, xRz\},$$

we have $\eta'_S(y)(f) = f(y) \geq a$. Next we show the converse. To prove the contrapositive, assume that $(x, y) \notin R$. By Definition 19, there is $f \in \mathrm{Cont}(S, \alpha)$ such that $(\Box_R f)(x) = 1$ and $f(y) \neq 1$. Then, $\eta'_S(x)(\Box_R f) \geq 1$ and $\eta'_S(y)(f) \not\geq 1$.

Now it remains to prove that η_S and η_S^{-1} satisfy the condition 2 in the arrow part of Definition 19, which follows immediately from the above facts.

The following is a Jónsson-Tarski-style duality for L-**ML**-algebras.

Theorem 6. *The category* L-MA *is dually equivalent to the category* L-RS *via the functors* RSpec(-) *and* MCont(-).

Proof. Let $\mathrm{Id}_{1'}$ denote the identity functor on L-MA and $\mathrm{Id}_{2'}$ denote the identity functor on L-RS. It is sufficient to show that there are natural isomorphisms $\epsilon' : \mathrm{Id}_{1'} \to \mathrm{MCont} \circ \mathrm{RSpec}$ and $\eta' : \mathrm{Id}_{2'} \to \mathrm{RSpec} \circ \mathrm{MCont}$. For an L-**ML**-algebra A, define ϵ'_A as in the proof of Theorem 4. For an object (S, α, R) in L-RS, define $\eta'_{(S,\alpha,R)}$ as in the proof of Theorem 5. Then it is straightforward to verify that η' and ϵ' are natural transformations. It follows from Theorem 4 and Theorem 5 that η' and ϵ' are natural isomorphisms.

The following is a compactness theorem for L-**ML**, which we prove using the compactness of the spectrum of an L-**ML**-algebra.

Theorem 7. *Let* $X \subset \mathbf{Form}_\Box$. *Assume that any finite subset of* X *is Kripke-satisfiable. Then,* X *is Kripke-satisfiable.*

Proof. We use the notions of L-filter, prime L-filter, and maximal L-filter (see [23, Definition 4 and Definition 5]).

Let A be the Lindenbaum algebra of L-**ML**. We may consider $X \subset A$. Then, $\{\langle x\rangle ; x \in X\}$ consists of clopen subsets of $\mathrm{Spec}_L(A)$. We show that $\{\langle x\rangle ; x \in X\}$ has finite intersection property. Since $\langle x\rangle \cap \langle y\rangle = \langle x \wedge y\rangle$ for $x, y \in A$, it suffices to show that $\langle x\rangle \neq \emptyset$ for any $x \in X$. Since $\{x\}$ is Kripke-satisfiable by assumption, $\mathrm{T}_1(x) \neq 0$. Let $\mathcal{F}$ be the set of those L-filters F of A such that $\mathrm{T}_1(x) \in F$ and $0 \notin F$. Then, $\{y \in A ; \mathrm{T}_1(x) \leq y\} \in \mathcal{F}$. By Zorn's lemma, we have a maximal element P in $\mathcal{F}$. Then, P is a maximal L-filter and so is a prime L-filter of A by [23, Proposition 3]. Define $v_P : A \to L$ by $v_P(z) = a \Leftrightarrow \mathrm{T}_a(z) \in P$ for $z \in A$. By [23, Proposition 4], v_P is a homomorphism with $v_P(x) = 1$, whence $v_P \in \langle x\rangle$. Thus, $\{\langle x\rangle ; x \in X\}$ has finite intersection property.

Since $\mathrm{Spec}_L(A)$ is compact and $\langle x\rangle$ is closed, we have $v \in \bigcap\{\langle x\rangle ; x \in X\}$. Consider the L-valued canonical model $(\mathrm{Spec}_L(A), R_\Box, e)$ of A. Then, $e(v, x) = v(x) = 1$ for any $x \in X$. Thus, X is Kripke-satisfiable.

4 Conclusions and Future Work

In this paper, we developed a natural duality for L-**VL**-algebras and a Jónsson-Tarski-style duality for L-**ML**-algebras. By applying the dualities, we obtained compactness theorems for L-**VL** and for L-**ML**, and the classification of equivalence classes of the categories L-VA's for finite distributive lattices L.

The dualities developed in this paper are essentially different from the ones mentioned as future work in [23, Section 4]. The dualities mentioned in [23] are based on the theory of canonical extensions ([15,14]), while the dualities in this paper are based on the theory of natural dualities ([7]). In other words, L-**VL** is considered as a many-valued logic in the dualities in this paper, while L-**VL** is considered as a modal logic in the dualities mentioned in [23], where note that the theory of canonical extensions can be considered as a general theory of dualities for modal-like logics (or lattices with operators). Let us explain the difference in more details in the following three paragraphs.

In the dualities mentioned in [23], we consider U_a's as modalities and equip the dual space of an L-**VL**-algebra with some canonical relations corresponding to each U_a for $a \in L$ (see [15, Subsection 2.3]), where recall that U_a's are interdefinable with T_a's. On the other hand, in the dualities in this paper, U_a's (or T_a's) are considered as the same kind of operations as the other operations of L-**VL**-algebras and the dual space $\mathrm{Spec}_L(A)$ of an L-**VL**-algebra A is equipped with no relation. This seems to be the most striking difference between the dualities mentioned in [23] and the dualities presented in this paper.

One of the most significant aspects of topological dualities is that we can understand the geometric meanings of logics or algebras by them. Since a duality becomes less geometric when a dual space is equipped with a relation, we may consider from the geometric viewpoint that it is better to equip a dual space with as less relations as possible and therefore the dualities developed in this paper are superior to the dualities mentioned in [23].

However, it also seems to be significant to see U_a's as modalities, i.e., L-**VL** as a multi-modal logic, and develop dualities for L-**VL**-algebras and L-**ML**-algebras in the way described above. The reasons are as follows. It seems interesting that a many-valued logic can be seen also as a modal logic. From the mathematical point of view, it confirms the applicability of the theory of canonical extensions, which is one of the most important duality theories along with the theory of natural dualities. From the philosophical point of view, it would contribute to our understanding of the notion of modality. Thus, our future work will be to develop dualities for L-**VL**-algebras and for L-**ML**-algebras by considering U_a's as modalities and using the theory of canonical extensions.

Finally, there is a remark on Theorem 3. Let L_1 and L_2 be finite distributive lattices. Theorem 3 implies that, even if L_1 and L_2 are not isomorphic, L_1-VA and L_2-VA can be categorically equivalent. This means that, even if L_1-**VL** and L_2-**VL** are not equivalent as logics, L_1-VA and L_2-VA can be categorically equivalent, which might contradict our intuition. By this fact, it seems that the identity of logics is strictly weaker (as a relation) than the equivalence of categories of Lindenbaum algebras of logics (for a discussion on the identity of logics, see [26]).

Acknowledgement. The author would like to thank Kentaro Sato for his suggesting a similar result to Theorem 3 for the category of algebras of Łukasiewicz n-valued logic.

References

1. Abramsky, S.: Domain theory in logical form. Ann. Pure Appl. Logic 51, 1–77 (1991)
2. Awodey, S.: Category theory. OUP (2006)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal logic. CUP (2001)
4. Bonsangue, M.M.: Topological duality in semantics. Electr. Notes Theor. Comput. Sci. 8 (1998)
5. Brink, C., Rewitzky, I.M.: A paradigm for program semantics: power structures and duality. CSLI Publications, Stanford (2001)
6. Burris, S., Sankappanavar, H.P.: A course in universal algebra. Springer, Heidelberg (1981)
7. Clark, D.M., Davey, B.A.: Natural dualities for the working algebraist. CUP (1998)
8. Connes, A.: Noncommutative geometry. Academic Press, London (1994)
9. Chagrov, A., Zakharyaschev, M.: Modal logic. OUP (1997)
10. Doran, R.S., Belfi, V.A.: Characterizations of C*-algebras; The Gelfand-Naimark theorems. Marcel Dekker Inc., New York (1986)
11. Grothendieck, A., Dieudonné, J.: Éléments de géométrie algébrique: I. Le langage des schémas. Publications Mathématiques de l'IHÉS 4, 225–228 (1960)
12. Hansoul, G.: A duality for Boolean algebras with operators. Algebra Universalis 17, 34–49 (1983)
13. Eleftheriou, P.E., Koutras, C.D.: Frame constructions, truth invariance and validity preservation in many-valued modal logic. J. Appl. Non-Classical Logics 15, 367–388 (2005)
14. Gehrke, M., Harding, J.: Bounded lattice expansions. J. Algebra 239, 345–371 (2001)
15. Gehrke, M., Nagahashi, H., Venema, Y.: A Sahlqvist theorem for distributive modal logic. Ann. Pure Appl. Logic 131, 65–102 (2005)
16. Fitting, M.C.: Many-valued modal logics. Fund. Inform. 15, 235–254 (1991)
17. Fitting, M.C.: Many-valued modal logics II. Fund. Inform. 17, 55–73 (1992)
18. Fitting, M.C.: Many-valued non-monotonic modal logics. In: Nerode, A., Taitslin, M.A. (eds.) LFCS 1992. LNCS, vol. 620, pp. 139–150. Springer, Heidelberg (1992)
19. Fitting, M.C.: Tableaus for many-valued modal logic. Studia Logica 55, 63–87 (1995)
20. Koutras, C.D., Zachos, S.: Many-valued reflexive autoepistemic logic. Logic Journal of the IGPL 8, 33–54 (2000)
21. Koutras, C.D., Peppas, P.: Weaker axioms, more ranges. Fund. Inform. 51, 297–310 (2002)
22. Koutras, C.D.: A catalog of weak many-valued modal axioms and their corresponding frame classes. J. Appl. Non-Classical Logics 13, 47–72 (2003)
23. Maruyama, Y.: Algebraic study of lattice-valued logic and lattice-valued modal logic. In: Ramanujam, R., Sarukkai, S. (eds.) ICLA 2009. LNCS (LNAI), vol. 5378, pp. 172–186. Springer, Heidelberg (2009)
24. Maruyama, Y.: The logic of common belief, revisited (in preparation)
25. Stone, M.H.: The representation of Boolean algebras. Bull. Amer. Math. Soc. 44, 807–816 (1938)
26. Straßburger, L.: What is a logic, and what is a proof? Logica Universalis 2nd edn., 135–152 (2007)
27. Teheux, B.: A duality for the algebras of a Łukasiewicz $n+1$-valued modal system. Studia Logica 87, 13–36 (2007)

An Independence Relation for Sets of Secrets

Sara Miner More and Pavel Naumov

Department of Mathematics and Computer Science
McDaniel College, Westminster, Maryland 21157, USA
{smore,pnaumov}@mcdaniel.edu

Abstract. A relation between two secrets, known in the literature as *nondeducibility*, was originally introduced by Sutherland. We extend it to a relation between sets of secrets that we call *independence*. This paper proposes a formal logical system for the independence relation, proves the completeness of the system with respect to a semantics of secrets, and shows that all axioms of the system are logically independent.

1 Introduction

In this paper we study interdependence between secrets. For example, if b_1, b_2, and b_3 are secrets with boolean values, then $b_1 \oplus b_2 \oplus b_3 = 0$ is an example of interdependence. If an interdependence between secrets is fixed and is publicly known, then knowledge of one secret may reveal something about the other secrets. In the above example, knowing the value of secret b_1 reveals whether or not secrets b_2 and b_3 are equal.

Let us now suppose that $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_k\}$ are two sets of secrets that are not interdependent. That is, knowledge of values $a_1, \dots, a_n$ reveals no information about values $b_1, \dots, b_k$. In this case we say that the sets of secrets A and B are *independent*. We will use the notation $A \parallel B$ to denote independence of A and B. If $n = k = 1$, then the independence predicate is essentially equivalent to the "no information flow" relation introduced by Sutherland [1].

In this work, we study properties of the independence predicate that are true regardless of the publicly-known interdependencies between secrets that may exist. For example, for any three secrets a, b, and c, if secrets a and b together reveal no information about secret c, then secret a alone will also reveal no information about secret c:

$$a, b \parallel c \to a \parallel c$$

A less obvious property of independence that can be expressed in propositional language and which is true regardless of the set of interdependencies that exist is:

$$a, b \parallel c \to (a \parallel b \to a \parallel b, c) \tag{1}$$

Below, we introduce a set of axioms for the independence predicate and prove the completeness of our logical system with respect to a semantics of secrets. In particular, property (1) above will follow from these axioms. We call this logical system *Logic of Secrets*.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 296–304, 2009.

Our work is related to the study of information flow. Most of the literature in this area, however, studies information flow from the language-based [2,3] or probabilistic [4,5] points of view. Historically ([6], page 185), one of the first attempts to capture independence in our sense was undertaken by Goguen and Meseguer [7] through their notion of *noninterference* between two computing devices. Later, Sutherland [1] introduced his *no information flow* relation, which is essentially our independence relation restricted to single-element sets. This relation has since become known in the literature as *nondeducibility*. Cohen [8] presented a related notion called *strong dependence*. Unlike nondeducibility, however, the strong dependence relation is not symmetric. More recently, Halpern and O'Neill [4,5] introduced f-secrecy to reason about multiparty protocols. In our notation, f-secrecy is a version of the nondeducibility predicate whose left or right side contains a certain function of the secret rather than the secret itself. However, all of these works focus on the application of the independence relation in the analysis of secure protocols, whereas the main focus of our work is on logical properties of the relation itself.

2 Semantics of Secrets

In this section we define a formal semantics for the independence relation.

Throughout the rest of this paper we assume that there is a fixed infinite set of "secret variables": $a, b, c, \ldots$. Intuitively, these variables can be viewed as names of secrets. A structure that serves as a model of the Logic of Secrets will be called a *protocol*. A protocol specifies names of the secret variables used, their possible values, and all publicly known interdependencies between secrets. The last of these is given as an explicit specification of all legitimate combinations of secret values, which we call "runs". Occasionally, we will refer to secret variables as just "secrets".

Definition 1. *A protocol is an arbitrary triple $\mathcal{P} = \langle \mathcal{S}, \mathcal{V}, \mathcal{R} \rangle$, where*

1. *$\mathcal{S}$ is a subset of the set of secret variables.*
2. *$\mathcal{V}$ is an arbitrary function that maps a secret variable $s \in \mathcal{S}$ into an arbitrary "set of values" of this secret $\mathcal{V}(s)$.*
3. *$\mathcal{R}$ is a set of functions, called runs of the protocol, such that each run r assigns a value $r(s) \in \mathcal{V}(s)$ to each secret variable $s \in \mathcal{S}$.*

For any protocol $\mathcal{P}$, by $\mathcal{R}(\mathcal{P})$ we mean the set of all runs of this protocol.

Definition 2. *A protocol $\mathcal{P} = \langle \mathcal{S}, \mathcal{V}, \mathcal{R} \rangle$ is finite if set $\mathcal{S}$ is finite and $\mathcal{V}(s)$ is finite for any $s \in \mathcal{S}$.*

In the following definition, and in the remainder of the paper, we write $f =_X g$ if $f(x) = g(x)$ for any $x \in X$.

Definition 3. *A set of secret variables $A \subseteq \mathcal{S}$ is independent from a set of secret variables $B \subseteq \mathcal{S}$ under protocol $\mathcal{P}$, if for any runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$ there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r =_A r_1$ and $r =_B r_2$.*

A special case of the independence predicate is the statement "the set of variables A is independent from the set of variables A". This statement, by definition, means that $r_1 =_A r_2$ for any runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$. In other words, for any $a \in A$, value $r(a)$ is the same for all runs $r \in \mathcal{R}(\mathcal{P})$. Thus, all secrets in A have fixed known values, and we will say that A is "public knowledge".

Definition 4. *The language of secrets consists of secret variables $a, b, c \ldots$, the independence predicate $\|$, implication $\to$, and false constant $\bot$. The set of formulas in this language is recursively defined as follows:*

1. *$\bot$ is a formula,*
2. *$X \parallel Y$ is a formula, for any two finite sets of secret variables X and Y,*
3. *if ϕ and ψ are formulas, then $\phi \to \psi$ is a formula.*

The language of secrets is similar to the universal fragment of propositional logic where $a_1, \ldots, a_n \parallel b_1, \ldots, b_k$ is a predicate of arity $n+k$. The difference, however, is that predicates in first order logic have a fixed arity, while our predicate $\|$ does not.

Definition 5. *We define a binary relation $\vDash$ between a protocol $\mathcal{P}$ and a formula ϕ by induction on the structural complexity of ϕ as follows:*

1. *$\mathcal{P} \nvDash \bot$,*
2. *$\mathcal{P} \vDash X \parallel Y$ if and only if X and Y are independent under $\mathcal{P}$,*
3. *$\mathcal{P} \vDash \phi \to \psi$ if and only if $\mathcal{P} \nvDash \phi$ or $\mathcal{P} \vDash \psi$.*

3 Logic of Secrets

Definition 6. *The Logic of Secrets is defined by the following axioms and inference rule:*

1. *All propositional tautologies in the language of secrets,*
2. *Empty Set Axiom: $\varnothing \parallel A$,*
3. *Monotonicity Axiom: $A, B \parallel C \to A \parallel C$,*
4. *Public Knowledge Axiom: $A \parallel A \to (B \parallel C \to A, B \parallel C)$,*
5. *Exchange Axiom: $A, B \parallel C, D \to (A \parallel B \to (D \parallel C \to A, C \parallel B, D))$,*
6. *Modus Ponens inference rule.*

Above and everywhere below, by A, B we mean $A \cup B$. As usual, we will write $X \vdash \phi$ if formula ϕ can be derived in the Logic of Secrets possibly using additional hypotheses from set X.

Lemma 1 (symmetry). *For any finite sets of secrets A and B,*

$$\vdash A \parallel B \to B \parallel A.$$

Proof. By Exchange Axiom, $\varnothing, A \parallel B, \varnothing \to (\varnothing \parallel A \to (\varnothing \parallel B \to \varnothing, B \parallel A, \varnothing))$. Taking into account Empty Set Axiom, $\varnothing A \parallel B, \varnothing \to \varnothing, B \parallel A, \varnothing$. Thus, $A \parallel B \to B \parallel A$.

As an example, let us now prove property (11) from these axioms. For convenience, we repeat the property below:

$$a, b \parallel c \rightarrow (a \parallel b \rightarrow a \parallel b, c)$$

By assuming $A = \{a\}$, $B = \{b\}$, $C = \varnothing$, and $D = \{c\}$ in Exchange Axiom, we get $a, b \parallel c \rightarrow (a \parallel b \rightarrow (c \parallel \varnothing \rightarrow a \parallel b, c))$. Thus, it will be sufficient to prove that $c \parallel \varnothing$. This, in turn, follows from Empty Set Axiom and Lemma 1.

Lemma 2. *If $X \vdash A \parallel B$, then $X \vdash A' \parallel B'$ for any $A' \subseteq A$ and $B' \subseteq B$.*

Proof. Follows from Monotonicity Axiom and Lemma 1.

4 Soundness

Theorem 1. *If $\vdash \phi$, then $\mathcal{P} \vDash \phi$ for any protocol $\mathcal{P}$.*

Proof. It will be sufficient to verify that $\mathcal{P} \vDash \phi$ for each axiom ϕ of the Logic of Secrets.

Empty Set Axiom. Consider any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$. Let $r = r_2$. It is easy to see that $r =_{\varnothing} r_1$ and $r =_A r_2$.

Monotonicity Axiom. Consider any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$. If $r =_{A,B} r_1$ and $r =_C r_2$, then $r =_A r_1$ and $r =_C r_2$.

Public Knowledge Axiom. Assume that $A \parallel A$ and $B \parallel C$. Consider any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$. By the assumption that $B \parallel C$, there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r =_B r_1$ and $r =_C r_2$. It will be sufficient to show that $r =_A r_1$. Indeed, by the assumption $A \parallel A$, there is a run $r' \in \mathcal{R}(\mathcal{P})$ such that $r =_A r' =_A r_1$. Therefore, $r =_A r_1$.

Exchange Axiom. Consider any two runs $r_1, r_2 \in \mathcal{R}(\mathcal{P})$. By the assumption that $A \parallel B$, there is a run $r_3 \in \mathcal{R}(\mathcal{P})$ such that $r_3 =_A r_1$ and $r_3 =_B r_2$. Since $D \parallel C$, there is a run $r_4 \in \mathcal{P}$ such that $r_4 =_D r_2$ and $r_4 =_C r_1$. Finally, by the assumption the $A, B \parallel C, D$, there is a run $r \in \mathcal{R}(\mathcal{P})$ such that $r =_{A,B} r_3$ and $r =_{C,D} r_4$. Thus, $r =_A r_3 =_A r_1$, $r =_C r_4 =_C r_1$, $r =_B r_3 =_B r_2$, and $r =_D r_4 =_D r_2$. Therefore, $r =_{A,C} r_1$ and $r =_{B,D} r_2$.

5 Completeness

Theorem 2. *If $\mathcal{P} \vDash \phi$ for any finite protocol $\mathcal{P}$, then $\vdash \phi$.*

The rest of the section contains the proof of this theorem. Assume that $\nvdash \phi$.

Definition 7. *Let $\mathcal{S}$ be the set of all secret variables appearing in ϕ.*

Definition 8. *Let Ψ be the minimal set that includes*

1. *all subformulas of ϕ and their negations,*
2. *$A \parallel B$ and $\neg(A \parallel B)$ for any $A, B \subseteq \mathcal{S}$*

Let X be a maximal consistent subset of Ψ that contains $\neg\phi$. We proceed now to define a finite protocol $\mathcal{P} = \langle \mathcal{S}, \mathcal{V}, \mathcal{R} \rangle$ such that $\mathcal{S}$ is the defined above set of secret variables. Later we will show that $\mathcal{P} \nvDash \phi$.

Definition 9. *For any secret $s \in \mathcal{S}$, we define set of values $\mathcal{V}(s)$ as follows:*

1. *if $X \vdash s \parallel s$, then $\mathcal{V}(s) = \{0\}$,*
2. *if $X \nvdash s \parallel s$, then $\mathcal{V}(s) = \{-1, 0, 1\}$.*

Next, we introduce terminology that allows us to define the set $\mathcal{R}$ of valid runs on protocol $\mathcal{P}$.

Definition 10. *A pair $(A, B) \in 2^{\mathcal{S}} \times 2^{\mathcal{S}}$ is called critical if*

1. *$X \nvdash A \parallel B$,*
2. *if $X \nvdash A' \parallel B'$, then $A = A'$ and $B = B'$, for any $A' \subseteq A$ and $B' \subseteq B$.*

Lemma 3. *For any pair $(A, B) \in 2^{\mathcal{S}} \times 2^{\mathcal{S}}$ such that $X \nvdash A \parallel B$, there is a critical pair (A', B') such that $A' \subseteq A$ and $B' \subseteq B$.*

Proof. Follows from finiteness of sets A and B.

Lemma 4. *If (C, D) is a critical pair, then $X \nvdash s \parallel s$ for any $s \in C \cup D$.*

Proof. Assume that $X \vdash s \parallel s$ for some $s \in C$. By Public Knowledge Axiom, $X \vdash C\backslash\{s\} \parallel D \to C \parallel D$. On the other hand, by the definition of critical pair, $X \nvdash C \parallel D$. Thus, $X \nvdash C\backslash\{s\} \parallel D$, which is a contradiction with the definition of critical pair. Therefore, $X \nvdash s \parallel s$. Case $s \in D$ is similar, due to Lemma 1.

Definition 11. *A run r is called void if there are sets of secrets C, D such that*

1. *pair (C, D) is critical,*
2. *$r(s) = 1$, for any $s \in C$,*
3. *$r(s) = -1$, for any $s \in D$.*

Definition 12. *Let $\mathcal{R}$ be the set of all runs that are not void.*

This concludes the definition of the finite protocol $\mathcal{P} = \langle \mathcal{S}, \mathcal{V}, \mathcal{R} \rangle$.

Lemma 5. *If $\mathcal{P} \vDash A \parallel B$, then $X \vdash A \parallel B$, for any $A, B \subseteq \mathcal{S}$.*

Proof. Assume that $X \nvdash A \parallel B$. By Lemma 3, there is a critical pair (A', B') such that $A' \subseteq A$ and $B' \subseteq B$. Consider runs r_+ and r_- such that for any secret s:

$$r_+(s) = \begin{cases} +1 & \text{if } X \nvdash s \parallel s \\ 0 & \text{otherwise} \end{cases}$$

$$r_-(s) = \begin{cases} -1 & \text{if } X \nvdash s \parallel s \\ 0 & \text{otherwise} \end{cases}$$

We will show that $r_+, r_- \in \mathcal{R}$. Let us start by showing that $r_+ \in \mathcal{R}$. Indeed, assume the opposite. Then there are $C, D \subseteq \mathcal{S}$ such that, taking into account Lemma 4 and Definition 11,

1. pair (C, D) is critical,
2. $+1 = r_+(s) = +1$, for any $s \in C$,
3. $+1 = r_+(s) = -1$, for any $s \in D$,

Note that the last statement implies that D is empty. Thus, by Empty Set Axiom, $\vdash D \parallel C$. By Lemma 1, $\vdash C \parallel D$. This contradicts the fact that (C, D) is a critical pair.

We now will prove that $r_- \in \mathcal{R}$. As in the previous case, assume the opposite. Hence, there are sets of secrets C, D such that, taking into account Lemma 4 and Definition 11,

1. pair (C, D) is critical,
2. $-1 = r_-(s) = +1$, for any $s \in C$,
3. $-1 = r_-(s) = -1$, for any $s \in D$,

Note that the second statement implies C is empty. Thus, by Empty Set Axiom, $\vdash C \parallel D$, which contradicts the fact that (C, D) is a critical pair.

We are ready to show that $\mathcal{P} \nvdash A \parallel B$. Indeed, by Definition 11, there is no run $r \in \mathcal{R}$ such that $\forall s \in A'\ (r(s) = +1)$ and $\forall s \in B'\ (r(s) = -1)$. Hence, there is no run $r \in \mathcal{R}$ such that $\forall s \in A'\ (r(s) = r_+(s))$ and $\forall s \in B'\ (r(s) = r_-(s))$. Finally, since $A' \subseteq A$ and $B' \subseteq B$, there is no run $r \in \mathcal{R}$ such that $\forall s \in A\ (r(s) = r_+(s))$ and $\forall s \in B\ (r(s) = r_-(s))$. Therefore, $\mathcal{P} \nvdash A \parallel B$.

Lemma 6. *If $X \vdash A \parallel B$, then $\mathcal{P} \vDash A \parallel B$.*

Proof. Assume that $X \vdash A \parallel B$. Consider any two runs $r_1, r_2 \in \mathcal{R}$. We need to find a run $r \in \mathcal{R}$ such that $\forall s \in A\ (r(s) = r_1(s))$ and $\forall s \in B\ (r(s) = r_2(s))$. Consider run r, defined as

$$r(s) = \begin{cases} r_1(s) & \text{if } s \in A \\ r_2(s) & \text{if } s \in B \\ 0 & \text{otherwise} \end{cases}$$

We will start by proving that run r is well-defined. For this, we need to show that $r_1(s) = r_2(s)$ if $s \in A \cap B$. Indeed, consider any $s \in A \cap B$. Note that $X \vdash A \parallel B$. Thus, by Lemma 2, $X \vdash s \parallel s$. Hence, by Definition 9, $\mathcal{V}(s) = \{0\}$. Therefore, $r_1(s) = r_2(s)$.

We now only need to show that $r \in \mathcal{R}$. In other words, we need to show that run r is not void. Assume the opposite. Hence, there are sets of secrets $C, D \subseteq \mathcal{S}$ such that

1. (C, D) is a critical pair,
2. $r(s) = +1$, for any $s \in C$,
3. $r(s) = -1$, for any $s \in D$.

Note that $r(s) = 0$ for any $s \notin A \cup B$. Thus,

$$C = (C \cap A) \cup (C \cap B) \tag{2}$$

$$D = (D \cap A) \cup (D \cap B) \tag{3}$$

Case 1: $(C \cap A, D \cap A) = (C, D)$. Thus, $C \subseteq A$ and $D \subseteq A$. Hence $r_1(s) = r(s) = +1$, for any $s \in C$, and $r_1(s) = r(s) = -1$, for any $s \in D$. Therefore, r_1 is void, which is a contradiction.

Case 2: $(C \cap B, D \cap B) = (C, D)$. Similar to Case 1.

Case 3: $(C \cap A, D \cap A) \neq (C, D)$ and $(C \cap B, D \cap B) \neq (C, D)$. Since (C, D) is a critical pair, these two statements imply that

$$X \vdash C \cap A \parallel D \cap A \tag{4}$$

and

$$X \vdash C \cap B \parallel D \cap B. \tag{5}$$

Note that by the assumption of the theorem, $X \vdash A \parallel B$. Thus, by Lemma 2,

$$X \vdash C \cap A, D \cap A \parallel C \cap B, D \cap B$$

By Exchange Axiom, using (4), (5), and Lemma 1,

$$X \vdash C \cap A, C \cap B \parallel D \cap A, D \cap B$$

Taking in to account (2) and (3), $X \vdash C \parallel D$, which contradicts the fact that the pair (C, D) is critical.

Lemma 7. *For any $\psi \in \Psi$, $\mathcal{P} \vDash \psi$ if and only if $\psi \in X$.*

Proof. We use induction on the structural complexity of ψ.

1. If $\psi \equiv \bot$, then $\mathcal{P} \nvDash \bot$ and, since X is consistent, $X \nvdash \bot$.
2. If $\psi \equiv \psi_1 \to \psi_2$, then $\mathcal{P} \nvDash \psi$ if and only if $\mathcal{P} \vDash \psi_1$ and $\mathcal{P} \nvDash \psi_2$. Thus, by the induction hypothesis, $\mathcal{P} \nvDash \psi$ if and only if $X \vdash \psi_1$ and $X \nvdash \psi_2$. Hence, since X is a maximal and consistent set of formulas, $\mathcal{P} \vDash \psi$ if and only if $\psi \in X$.
3. $\psi \equiv A \parallel B$. See Lemma 5 and Lemma 6.

Finally, we note that Theorem 2 follows from the previous lemma.

6 Axiom Independence

In this section we will prove that each of the axioms of the Logic of Secrets is independent from the other axioms. This is done by defining non-standard semantics for the independence predicate.

Theorem 3. *Empty Set Axiom is not provable from the other axioms.*

Proof. Consider a new semantics of the independence predicate under which $A \parallel B$ is false for all sets of secret variables A and B. Under this non-standard

semantics, Empty Set Axiom is false, but Monotonicity, Public Knowledge, and Exchange Axioms are true. Therefore, Empty Set Axiom is independent from the other axioms.

Theorem 4. *Monotonicity Axiom is not provable from the other axioms.*

Proof. Fix an arbitrary secret variable s_0. Consider a new semantics of the independence predicate under which $A \parallel B$ is true if and only if at least one of the following conditions is true:

1. A is empty,
2. B is empty,
3. $s_0 \in A \cup B$.

Let us show that this definition satisfies Empty Set, Public Knowledge, and Exchange axioms, and does not satisfy Monotonicity axiom.

Empty Set Axiom. $A \parallel \varnothing$ because $\varnothing$ is an empty set.

Public Knowledge Axiom. Assume that $A \parallel A$ and $B \parallel C$. The first of these statements implies that either A is empty or $s_0 \in A$. If A is empty, then $A, B = B$. Hence, $B \parallel C$ implies $A, B \parallel C$. Suppose $s_0 \in A$. Thus, $s_0 \in A \cup B \cup C$, and therefore, $A, B \parallel C$.

Exchange Axiom. Assume that $A, B \parallel C, D$ as well as $A \parallel B$ and $D \parallel C$. If $s_0 \in A \cup B \cup C \cup D$, then $A, C \parallel B, D$ is true. Suppose that $s \notin A \cup B \cup C \cup D$. Thus, $A \parallel B$ and $D \parallel C$ imply that one set out of A and B and one set out of C and D are empty. If empty sets are A and C or B and D, then $A, C \parallel B, D$ is true. So, it will be sufficient to consider the case when A and D are empty or B and C are empty.

1. First, consider the case where A and D are empty. Assumption $A, B \parallel C, D$ implies that $B \parallel C$. Hence, either B or C is empty. Therefore, either $B \cup D$ or $A \cup C$ is empty. Thus, $A, C \parallel B, D$.
2. Second, consider the case where B and C are empty. Assumption $A, B \parallel C, D$ implies that $A \parallel D$. Hence, either A or D is empty. Therefore, either $A \cup C$ or $B \cup D$ is empty. Thus, $A, C \parallel B, D$.

Monotonicity Axiom. Let t and u be secret variables different from variable s_0. Consider any protocol $\mathcal{P}$ and sets $A = \{t\}$, $B = \{s_0\}$, and $C = \{u\}$. By definition, $\mathcal{P} \vDash A, B \parallel C$, but $\mathcal{P} \nvDash A \parallel C$.

Theorem 5. *Public Knowledge Axiom is not provable from the other axioms.*

Proof. Consider a new semantics of the independence predicate under which secret variables are interpreted as nodes of a certain undirected graph. Independence predicate $A \parallel B$ is true if and only if there is *no* crossing edge that connects a node from set A with a node from set B. It is easy to see that Empty Set Axiom and Monotonicity Axiom are true under this interpretation.

Exchange Axiom. Suppose that $A, B \parallel C, D$, as well as $A \parallel B$ and $D \parallel C$. We will need to show that $A, C \parallel B, D$. Assume the opposite: there is a crossing edge e from $A \cup C$ to $B \cup D$. There are four cases to consider: (a) if e goes from

A to B, then $A \parallel B$ is false, (b) if e goes from A to D, then $A, B \parallel C, D$ is false, (c) if e goes from C to B, then $A, B \parallel C, D$ is false, (d) if e goes from C to D, then $D \parallel C$ is false.

Public Knowledge Axiom. Finally, we will show that there is a graph G and sets of nodes A, B, and C, for which $A \parallel A \rightarrow (B \parallel C \rightarrow A, B \parallel C)$ is false. Let graph G consist of only three nodes a, b, and c. Assume that (a, c) is the only edge of this graph. Note that $a \parallel a$ and $b \parallel b$ are true, but $a, b \parallel c$ is false.

Theorem 6. *Secret Exchange Axiom is not provable from the other axioms.*

Proof. Consider a non-standard semantics for independence predicate under which $A \parallel B$ stands for "set A is empty". It is easy to see that Empty Set Axiom, Monotonicity Axiom, and Public Knowledge Axiom are true under this interpretation. At the same time, if sets A, B, and D are empty and set C is not, then Exchange Axiom is false.

7 Conclusion

In this paper, we have introduced a logical system that describes properties of independence between two sets of secret variables. Naturally, one can ask about an independence predicate for three or more sets of secret variables. For example, an independence predicate for three sets A, B, and C could be defined as

$$A \parallel B \parallel C \quad \Longleftrightarrow \quad \forall r_1, r_2, r_3\, \exists r\, (r =_A r_1 \wedge r =_B r_2 \wedge r =_C r_3).$$

We conclude with the observation that independence predicates that have more than two sets of arguments can be expressed through the two-argument independence predicate studied in this paper. For example, it can be shown that $A \parallel B \parallel C$ is logically equivalent to the conjunction $(A \parallel B) \wedge (A, B \parallel C)$.

References

1. Sutherland, D.: A model of information. In: Proceedings of Ninth National Computer Security Conference, pp. 175–183 (1986)
2. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications 21(1), 5–19 (2003)
3. Amtoft, T., Banerjee, A.: A logic for information flow analysis with an application to forward slicing of simple imperative programs. Sci. Comput. Program. 64(1), 3–28 (2007)
4. Halpern, J., O'Neill, K.: Secrecy in multiagent systems. In: Proceedings of the Fifteenth IEEE Computer Security Foundations Workshop, pp. 32–46 (2002)
5. Halpern, J.Y., O'Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. 12(1), 1–47 (2008)
6. MacKenzie, D.: Mechanizing Proof: Computing, Risk, and Trust. MIT Press, Cambridge (2004)
7. Goguen, J.A., Meseguer, J.: Security policies and security models. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 11–20 (1982)
8. Cohen, E.: Information transmission in computational systems. In: Proceedings of Sixth ACM Symposium on Operating Systems Principles, Association for Computing Machinery, pp. 113–139 (1977)

Expressing Extension-Based Semantics Based on Stratified Minimal Models

Juan Carlos Nieves[1], Mauricio Osorio[2], and Claudia Zepeda[3]

[1] Universitat Politècnica de Catalunya
Software Department (LSI)
c/Jordi Girona 1-3, E08034, Barcelona, Spain
jcnieves@lsi.upc.edu
[2] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@googlemail.com
[3] Benemérita Universidad Atónoma de Puebla
Facultad de Ciencias de la Computación,
Puebla, Puebla, México
czepedac@gmail.com

Abstract. Extension-based argumentation semantics is a successful approach for performing non-monotonic reasoning based on argumentation theory. An interesting property of some extension-based argumentation semantics is that these semantics can be characterized in terms of logic programming semantics. In this paper, we present novel results in this topic. In particular, we show that one can induce an argumentation semantics (that we call Stratified Argumentation Semantics) based on a logic programming semantics that is based on stratified minimal models. We show that the stratified argumentation semantics overcome some problems of extension-based argumentation semantics based on admissible sets and we show that it coincides with the argumentation semantics CF2.

Keywords: Non-monotonic reasoning, extension-based argumentation semantics and logic programming.

1 Introduction

Argumentation theory has become an increasingly important and exciting research topic in Artificial Intelligence (AI), with research activities ranging from developing theoretical models, prototype implementations, and application studies [3]. The main purpose of argumentation theory is to study the fundamental mechanism, humans use in argumentation, and to explore ways to implement this mechanism on computers.

Dung's approach, presented in [6], is a unifying framework which has played an influential role on argumentation research and AI. This approach is mainly orientated to manage the interaction of arguments. The interaction of the arguments is supported by four extension-based argumentation semantics: *stable*

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 305–319, 2009.

a) **b)**

Fig. 1. In **a)**, it is presented the graph representation of the argumentation framework: $\langle\{a, b\}, \{(a, a), (a, b)\}\rangle$. In **b)**, it is presented the graph representation of the argumentation framework: $\langle\{a, b, c, d, e\}, \{(a, c), (c, b), (b, a), (a, d), (c, d), (b, d), (d, e)\}\rangle$.

semantics, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of these semantics is the *acceptability of the arguments*. It is worth mentioning that although these argumentation semantics represents different pattern of selection of arguments, all these argumentation semantics are based on the concept of admissible set.

An important point to remark *w.r.t.* the argumentation semantics based on admissible sets is that these semantics exhibit a variety of problems which have been illustrated in the literature [17,2,3]. For instance, let AF be the argumentation framework which appears in Figure 1-a. In this AF there are two arguments: a and b. The arrows in the figure represent conflicts between the arguments. We can see that the argument a is attacked by itself and the argument b is attacked by the argument a. Some authors as Prakken and Vreeswijk [17] suggest in a intuitive way, that one can expect that the argument b can be considered as an acceptable argument since it is attacked by the argument a which is attacked by itself. However, none of the argumentation semantics suggested by Dung is able to infer the argument b as acceptable.

Another interesting argumentation framework which has been commented on literature [17,2] is presented in Figure 1-b.

Some authors, as Prakken and Vreeswijk [17], Baroni *et al* [2], suggest that the argument e can be considered as an acceptable argument since it is attacked by the argument d which is attacked by three arguments: a, b, c. Observe that the arguments a, b and c form a cyclic of attacks.

We can recognize two major branches for improving Dung's approach. On the one hand, we can take advantage of graph theory; on the other hand, we can take advantage of logic programming with negation as failure.

With respect to graph theory, the approach suggested by Baroni *et al*, in [2] is maybe the most general solution defined until now for improving Dung's approach. This approach is based on a solid concept in graph theory which is a *strongly connected component* (SCC). Based on this concept, Baroni *et al*, describe a recursive approach for generating new argumentation semantics. For instance, the argumentation semantics CF2 suggested in [2] is able to infer the argument b as an acceptable argument of the AF of Figure 1-a. Also CF2 regards the argument e as an acceptable argument from the AF of Figure 1-b.

Since Dung's approach was introduced in [6], it was viewed as a special form of logic programming with *negation as failure*. For instance, in [6] it was proved that the grounded semantics can be characterized by the well-founded semantics [8], and the stable argumentation semantics can be characterized by the stable model semantics [9]. Also in [4], it was proved that the preferred semantics can be characterized by the p-stable semantics [16]. In fact, the preferred semantics can be also characterized by the minimal models and the stable models of a logic program [14]. By regarding an argumentation framework in terms of logic programs, it has been shown that one can construct intermediate argumentation semantics between the grounded and preferred semantics [12]. Also it is possible to define extensions of the preferred semantics [15].

When we have a logic program which represents an argumentation framework, it is natural to think that we can split this program into subprograms where each subprogram could represents a part of an argumentation framework. The idea of splitting a logic program into its component, in order to define logic programming semantics, has been explored by some authors in logic programming [5]. For instance, by splitting a logic program, Dix and Müller in [5] combine ideas of the stable model semantics and the well-founded semantics in order to define a skeptical logic programming semantics which satisfies the property of relevance and the general principle of partial evaluation.

In this paper, we are going to explore the idea of splitting a logic program into its components in order to achieve two main objectives:

1. To explore the definition of candidate argumentation semantics in terms of logic programming semantics. In particular, we define an extension-based argumentation semantics, that we call *stratified argumentation semantics*. This semantics will be induced by the stratified minimal model semantics. We will show that this new argumentation semantics coincides with CF2 which is considered as the most acceptable argumentation semantics introduced in [2].
2. To introduce a recursive construction which define a new logic programming semantics, that we call *stratified minimal models semantics*. Based on the construction of this semantics, we will show that there exists a family of logic programming semantics that are always defined and satisfy the property of relevance.

The rest of the paper is divided as follows: In §2, we present some basic concepts *w.r.t.* logic programming and argumentation theory. In §3, we define the stratified minimal model semantics and introduce our first main theorem. In §4, we introduce the stratified argumentation semantics and present our second main theorem of this paper. Finally in the last section, we present our conclusions.

2 Background

In this section, we define the syntax of the logic programs that we will use in this paper and some basic concepts of logic programming semantics and argumentation semantics.

2.1 Syntax and Some Operations

A signature $\mathcal{L}$ is a finite set of elements that we call atoms. A *literal* is either an atom a, called *positive literal*; or the negation of an atom $\neg a$, called *negative literal*. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of atoms $\{\neg a_1, ..., \neg a_n\}$. A *normal* clause, C, is a clause of the form

$$a \leftarrow b_1 \wedge \ldots \wedge b_n \wedge \neg b_{n+1} \wedge \ldots \wedge \neg b_{n+m}$$

where a and each of the b_i are atoms for $1 \leq i \leq n+m$. In a slight abuse of notation we will denote such a clause by the formula $a \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ where the set $\{b_1, \ldots, b_n\}$ will be denoted by $\mathcal{B}^+$, and the set $\{b_{n+1}, \ldots, b_{n+m}\}$ will be denoted by $\mathcal{B}^-$. We define a *normal program* P, as a finite set of normal clauses. If the body of a normal clause is empty, then the clause is known as a *fact* and can be denoted just by: $a \leftarrow$.

We write $\mathcal{L}_P$, to denote the set of atoms that appear in the clauses of P. We denote by $HEAD(P)$ the set $\{a | a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^- \in P\}$.

A program P induces a notion of *dependency* between atoms from $\mathcal{L}_P$. We say that a *depends immediately on* b, if and only if, b appears in the body of a clause in P, such that a appears in its head. The two place relation *depends on* is the transitive closure of *depends immediately on*. The set of dependencies of an atom x, denoted by *dependencies-of*(x), corresponds to the set $\{a \mid x \textit{ depends on } a\}$. We define an equivalence relation $\equiv$ between atoms of $\mathcal{L}_P$ as follows: $a \equiv b$ if and only if $a = b$ or (a *depends on* b and b *depends on* a). We write $[a]$ to denote the equivalent class induced by the atom a.

Example 1. Let us consider the following normal program,

$$S = \{e \leftarrow e,\ \ c \leftarrow c,\ \ a \leftarrow \neg b \wedge c,\ \ b \leftarrow \neg a \wedge \neg e,\ \ d \leftarrow b\}.$$

The dependency relations between the atoms of $\mathcal{L}_S$ are as follows: *dependencies-of*$(a) = \{a, b, c, e\}$; *dependencies-of*$(b) = \{a, b, c, e\}$; *dependencies-of*$(c) = \{c\}$; *dependencies-of*$(d) = \{a, b, c, e\}$; and *dependencies-of*$(e) = \{e\}$. We can also see that, $[a] = [b] = \{a, b\}$, $[d] = \{d\}$, $[c] = \{c\}$, and $[e] = \{e\}$.

We take $<_P$ to denote the strict partial order induced by $\equiv$ on its equivalent classes. Hence, $[a] <_P [b]$, if and only if, b *depends-on* a and $[a]$ is not equal to $[b]$. By considering the relation $<_P$, each atom of $\mathcal{L}_P$ is assigned an order as follows:

- An atom a *is of order* 0, if $[a]$ is minimal in $<_P$.
- An atom a *is of order* $n+1$, if n is the maximal order of the atoms on which a depends.

We say that a program P is of order n, if n is the maximum order of its atoms. We can also break a program P of order n into the disjoint union of programs P_i with $0 \leq i \leq n$, such that P_i is the set of rules for which the head of each clause is of order i (*w.r.t.* P). We say that $P_0, \ldots, P_n$ are the *relevant modules* of P.

Example 2. By considering the equivalent classes of the program S in Example 1, the following relations hold: $\{c, e\} <_S \{a, b\} <_S \{d\}$. We also can see that: a is of order 1, d is of order 2, b is of order 1, e is of order 0, and c is of order 0. This means that S is a program of order 2.

The following table illustrates how the program S can be broken into the disjoint union of the following relevant modules S_0, S_1, S_2:

S	S_0	S_1	S_2
$e \leftarrow e.$	$e \leftarrow e.$		
$c \leftarrow c.$	$c \leftarrow c.$		
$a \leftarrow \neg b \wedge c.$		$a \leftarrow \neg b \wedge c.$	
$b \leftarrow \neg a \wedge \neg e.$		$b \leftarrow \neg a \wedge \neg e.$	
$d \leftarrow b.$			$d \leftarrow b.$

Now we introduce a single reduction for any normal program. The idea of this reduction is to remove from a normal program any atom which has already fixed to some true value. In fact, this reduction is based on a pair of sets of atoms $\langle T; F\rangle$ such that the set T contains the atoms which can be considered as true and the set F contains the atoms which can be considered as false. Formally, this reduction is defined as follows:

Let $A = \langle T; F\rangle$ be a pair of sets of atoms. The reduction $R(P, A)$ is obtained by 4 steps:

1. We replace every atom x that occurs in the bodies of P by 1 if $x \in T$, and we replace every atom x that occurs in the bodies of P by 0 if $x \in F$;
2. we replace every occurrence of $\neg 1$ by 0 and $\neg\ 0$ by 1;
3. every clause with a 0 in its body is removed;
4. finally we remove every occurrence of 1 in the body of the clauses.

We want to point out that this reduction does not coincide with the Gelfond-Lifschitz reduction [9].

Example 3. Let us consider the normal program S of Example 1. Let P be the normal program $S \setminus S_0$, and let A be the pair of sets of atoms $\langle\{c\};\{e\}\rangle$. This means that we obtain the following programs:

P:
$a \leftarrow \neg b \wedge c.$
$b \leftarrow \neg a \wedge \neg e.$
$d \leftarrow b.$

$R(P, A)$:
$a \leftarrow \neg b.$
$b \leftarrow \neg a.$
$d \leftarrow b.$

2.2 Semantics

From now on, we assume that the reader is familiar with the single notion of *minimal model*. In order to illustrate this basic notion, let P be the normal program $\{a \leftarrow \neg b, \quad b \leftarrow \neg a, \quad a \leftarrow \neg c, \quad c \leftarrow \neg a\}$. As we can see, P has five models: $\{a\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b\}$, $\{a, b, c\}$; however, P has just two minimal models: $\{b, c\}$, $\{a\}$. We will denote by $MM(P)$ the set of all the minimal models of a given logic program P. Usually MM is called *minimal model semantics*.

A semantics SEM is a mapping from the class of all programs into the power-set of the set of (2-valued) models. SEM assigns to every program P a (possible empty) set of (2-valued) models of P. If $SEM(P) = \emptyset$, then we informally say that SEM is undefined for P.

Given a set of interpretations Q and a signature $\mathcal{L}$, we define Q *restricted to* $\mathcal{L}$ as $\{M \cap \mathcal{L} \mid M \in Q\}$. For instance, let Q be $\{\{a, c\}, \{c, d\}\}$ and $\mathcal{L}$ be $\{c, d, e\}$, hence Q *restricted to* $\mathcal{L}$ is $\{\{c\}, \{c, d\}\}$.

Let P be a program and $P_0, \ldots, P_n$ its relevant modules. We say that a semantics S satisfies the property of relevance if for every i, $0 \leq i \leq n$, $S(P_0 \cup \ldots \cup P_i) = S(P)$ restricted to $\mathcal{L}_{P_0 \cup \ldots \cup P_i}$.

2.3 Argumentation Basics

Now, we present some basic concepts with respect to extended-based argumentation semantics. The first concept that we consider is the one of *argumentation framework*. An argumentation framework captures the relationships between the arguments.

Definition 1. *[6] An argumentation framework is a pair* $AF = \langle AR, \text{attacks} \rangle$*, where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR*,* i.e. attacks $\subseteq AR \times AR$*. We write* $\mathcal{AF}_{AR}$ *to denote the set of all the argumentation frameworks defined over* AR*.*

We say that *a attacks b* (or *b* is attacked by *a*) if $(a, b) \in attacks$ holds. Usually an extension-based argumentation semantics S_{Arg} is applied to an argumentation framework AF in order to infer sets of acceptable arguments from AF. An extension-based argumentation semantics S_{Arg} is a function from $\mathcal{AF}_{AR}$ to 2^{AR}. S_{Arg} can be regarded as a pattern of selection of sets of arguments from a given argumentation framework AF.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, we will say that an argument $a \in AR$ is acceptable, if $a \in E$ such that $E \in S_{Arg}(AF)$.

3 Stratified Minimal Model Semantics

In this section, we introduce a constructive logic programming semantics, called *stratified minimal model semantics*, which is based on minimal models. This semantics has some interesting properties as: it satisfies the property of relevance, and it agrees with the stable model semantics for the well-known class of stratified logic programs (the proof of this property can be found in [12,13]).

In order to define the stratified minimal model semantics MM^r, we define the operator $*$ and the function $freeTaut$ as follows:

- Given Q and L both sets of interpretations, we define $Q * L := \{M_1 \cup M_2 \mid M_1 \in Q, M_2 \in L\}$.
- Given a logic program P, $freeTaut$ denotes a function which removes from P any tautology.

The idea of the function $freeTaut$ is to remove any clause which is equivalent to a tautology in classical logic.

Definition 2. *Given a normal logic program P, we define the* sstratified minimal model semantics MM^r *as follows:* $MM^r(P) = MM^r_c(freeTaut(P) \cup \{x \leftarrow x \mid x \in \mathcal{L}_P \setminus HEAD(P)\}$ *such that* $MM^r_c(P)$ *is defined as follows:*

1. *if P is of order 0, $MM^r_c(P) = MM(P)$.*
2. *if P is of order $n > 0$, $MM^r_c(P) = \bigcup_{M \in MM(P_0)} \{M\} * MM^r_c(R(Q, A))$ where $Q = P \setminus P_0$ and $A = \langle M; \mathcal{L}_{P_0} \setminus M \rangle$.*

We call a model in $MM^r(P)$ a stratified minimal model *of P.*

Observe that the definition of the stratified minimal model semantics is based on a recursive construction where the base case is the application of MM. It is not difficult to see that if one changes MM by any other logic programming semantics S, as the stable model semantics, one is able to construct a relevant version of the given logic programming semantics (see [12,13] for details).

In order to introduce an important theorem of this paper, let us introduce some concepts. We say that a normal program P is basic if every atom x that belongs to $\mathcal{L}_P$, then x occurs as a fact in P. We say that a logic programming semantics SEM is *defined for basic programs*, if for every basic normal program P then $SEM(P)$ is defined.

The following theorem shows that there exists a family of logic programming semantics that are always defined and satisfy the property of relevance.

Theorem 1. *For each semantics SEM that is defined for basic programs, there exists a semantics SEM' that satisfies the following:*

1. *For every normal program P, $SEM'(P)$ is defined.*
2. *SEM' is relevant.*
3. *SEM' is invariant under adding tautologies.*

An instantiation of SEM and SEM' of Theorem 1 are the semantics MM and MM^r respectively. Observe that essentially this theorem is suggesting that given any logic programming semantics SEM, such as MM, that is defined for basic program, one can construct a *relative similar semantic* SEM', such as MM^r, to SEM satisfying the three properties described in this Theorem 1.

4 Stratified Argumentation Semantics

In this section, we show that by considering the stratified minimal model semantics, one can induce an argumentation semantics. In fact, we show that this new argumentation semantics will take advantage of the properties of the stratified minimal model semantics.

As the stratified minimal model semantics is a semantics for logic programs, we require a function mapping able to construct a logic program from an argumentation framework. Hence, let us introduce a simple mapping to regard an

argumentation framework as a normal logic program. In this mapping, we use the predicates $d(x)$, $a(x)$. The intended meaning of $d(x)$ is: "the argument x is defeated" (this means that the argument x is attacked by an acceptable argument), and the intended meaning of $a(X)$ is that the argument X is accepted.

Definition 3. *Let $AF = \langle AR, attacks\rangle$ be an argumentation framework, $P^1_{AF} = \{d(a) \leftarrow \neg d(b_1), \ldots, d(a) \leftarrow \neg d(b_n) \mid a \in AR$ and $\{b_1, \ldots, b_n\} = \{b_i \in AR \mid (b_i, a) \in attacks\}\}$; and $P^2_{AF} = \bigcup_{a\in AR}\{a(a) \leftarrow \neg d(a)\}$. We define: $P_{AF} = P^1_{AF} \cup P^2_{AF}$.*

The intended meaning of the clauses of the form $d(a) \leftarrow \neg d(b_i)$, $1 \leq i \leq n$, is that an argument a will be defeated when anyone of its adversaries b_i is not defeated. Observe that, essentially, P^1_{AF} is capturing the basic principle of *conflict-freeness* (this means that any set of acceptable argument will not contain two arguments which attack each other). The idea P^2_{AF} is just to infer that any argument a that is not defeated is accepted.

Example 4. Let AF be the argumentation framework of Figure 1-b. We can see that $P_{AF} = P^1_{AF} \cup P^2_{AF}$ is:

P^1_{AF} :

$d(a) \leftarrow \neg d(b).$
$d(b) \leftarrow \neg d(c).$
$d(c) \leftarrow \neg d(a).$
$d(d) \leftarrow \neg d(a).$
$d(d) \leftarrow \neg d(b).$
$d(d) \leftarrow \neg d(c).$
$d(e) \leftarrow \neg d(d).$

P^2_{AF} :

$a(a) \leftarrow \neg d(a).$
$a(b) \leftarrow \neg d(b).$
$a(c) \leftarrow \neg d(c).$
$a(d) \leftarrow \neg d(d).$
$a(e) \leftarrow \neg d(e).$

Two relevant properties of the mapping P_{AF} are that the stable models of P_{AF} characterize the stable argumentation semantics and the well founded model of P_{AF} characterizes the grounded semantics [12].

Once we have defined a mapping from an argumentation framework into logic programs, we are going to define a candidate argumentation semantics which is induced by the stratified minimal model semantics.

Definition 4. *Given an argumentation framework A, we define a stratified extension of AF as follows: A_m is a stratified extension of AF if exists a stratified minimal model M of P_{AF} such that $A_m = \{x | a(x) \in M\}$. We write $MM^r_{Arg}(AF)$ to denote the set of stratified extensions of AF. This set of stratified extensions is called stratified argumentation semantics.*

In order to illustrate the stratified argumentation semantics, we are going to presents some examples.

Example 5. Let AF be the argumentation framework of Figure 1-b and P_{AF} be the normal program defined in Example 4. In order to infer the stratified argumentation semantics, we infer the stratified minimal models of P_{AF}. As we can see P_{AF} has three stratified minimal models : $\{d(a), d(b), d(d), a(c), a(e)\}\{d(b), d(c), d(d), a(a), a(e)\}\{d(a), d(c), d(d), a(b), a(e)\}$, this means that AF has three

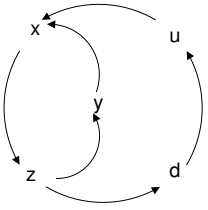


Fig. 2. Graph representation of $AF = \langle\{x, y, z, u, d\}, \{(x, z),\ (z, y),\ (y, x),\ (u, x),\ (z, d), (d, u)\}\rangle$

stratified extensions which are: $\{c, e\}$, $\{a, e\}$ and $\{b, e\}$. Observe that the stratified argumentation semantics coincides with the argumentation semantics CF2.

Let us consider another example.

Example 6. Let us consider the argumentation framework of Figure 2. It is not difficult to obtain its $P_{AF} = P^1_{AF} \cup P^2_{AF}$ where P^1_{AF} and P^2_{AF} correspond to the following programs:

P^1_{AF} :

$d(x) \leftarrow \neg d(y).$
$d(y) \leftarrow \neg d(z).$
$d(z) \leftarrow \neg d(x).$
$d(x) \leftarrow \neg d(u).$
$d(d) \leftarrow \neg d(z).$
$d(u) \leftarrow \neg d(d).$

P^2_{AF} :

$a(x) \leftarrow \neg d(x).$
$a(y) \leftarrow \neg d(y).$
$a(z) \leftarrow \neg d(z).$
$a(d) \leftarrow \neg d(d).$
$a(u) \leftarrow \neg d(u).$

Now let us compute the argumentation semantics MM^r_{Arg}. Since $MM^r(P_{AF})$= $\{\ \{d(y), d(z), d(u), a(x), a(d)\},\ \{d(x), d(z), d(d), a(y), a(u)\},\ \{d(u), d(x), d(z), a(y), a(d)\},\ \{d(x), d(y), d(d), a(z), a(u)\}\ \}$ then, $MM^r_{Arg}(AF) = \{\ \{x, d\}, \{y, u\}, \{y, d\}, \{z, u\}\ \}$. Notice that MM^r_{Arg} coincides with the argumentation semantics CF2.

We are going to present our second main theorem of this paper. This theorem formalizes that the stratified argumentation semantics and the argumentation semantics CF2 coincide.

Theorem 2. *Given an argumentation framework $AF = \langle AR, Attacks\rangle$, and $E \in AR$, $E \in MM^r_{Arg}(AF)$ if and only if $E \in CF2(AF)$.*

As final result of this paper, we show an important result *w.r.t.* the decision problem of knowing if a set of arguments is a stratified extension.

Lemma 1. *Given an argumentation framework $AF = \langle AR, Attacks\rangle$ and a set of argument $E \subseteq AR$, the decision problem of knowing if E is a stratified extension of AF is polynomial time computable.*

Observe that by this lemma and Theorem 2, one can infer that the decision problem of knowing if a set of arguments belongs to CF2 is polynomial time computable. Recall that on the other hand, the corresponding complexity decision problem for the preferred semantics is CO-NP-Complete [7].

5 Conclusions

It is well-accepted that extension-based argumentation semantics is a promising approach for performing non-monotonic reasoning. However, since in the literature of argumentation has been exhibited a variety of problems of some of the existing argumentation semantics, nowadays it has increased the number of new argumentation semantics in the context of Dung's argumentation approach. We have to recognize that many of these new argumentation semantics are only motivated by particular examples, and also these introduced argumentation semantics lack of logic foundations. In this paper, we show that one can induce novel argumentation semantics by considering logic programming semantics. In particular, we introduce a novel argumentation semantics (stratified argumentation semantics) based on a new logic programming semantics (stratified minimal model semantics). In fact, we show that the stratified argumentation semantics coincides with the argumentation semantics CF2 which was introduced in terms of graph theory's terms (Theorem 2). It is worth mentioning that the stratified argumentation semantics is just one of the multiples candidate argumentation semantics that can be induced by the family of logic programming semantics identified by Theorem 1 (for more details about other new candidate argumentation semantics see [12,13]).

An important property of the stratified argumentation semantics is that the decision problem of knowing if a set of arguments is a stratified extension is polynomial time computable. This means that this semantics is computationally less expensive than the preferred semantics. This result also suggests that the decision problem of knowing if a set of arguments belongs to CF2 is polynomial time computable. We believe that the study of argumentation semantics in terms of logic programming semantics could help to explore the non-monotonic properties of the argumentation semantics. The study of non-monotonic properties of an argumentation semantics could suggests some guidelines in order to find suitable argumentation semantics for the applications of these to real domains.

Acknowledgement

We are grateful to anonymous referees for their useful comments. We would like to acknowledge support from the EC founded project ALIVE (FP7-IST-215890). The views expressed in this paper are not necessarily those of the ALIVE consortium.

References

1. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications, 2nd edn. Springer, Heidelberg (2008)
2. Baroni, P., Giacomin, M., Guida, G.: SCC-recursiveness: a general schema for argumentation semantics. Artificial Intelligence 168, 162–210 (2005)
3. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence 171(10-15), 619–641 (2007)

4. Carballido, J.L., Nieves, J.C., Osorio, M.: Inferring Preferred Extensions by Pstable Semantics. Iberoamerican Journal of Artificial Intelligence (Inteligencia Artificial) 13(41), 38–53 (2009)
5. Dix, J., Müller, M.: Partial evaluation and relevance for approximations of stable semantics. In: Raś, Z.W., Zemankova, M. (eds.) ISMIS 1994. LNCS, vol. 869, pp. 511–520. Springer, Heidelberg (1994)
6. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–358 (1995)
7. Dunne, P.E., Bench-Capon, T.J.: Coherence in finite argument systems. Artificial Intelligence 141(1), 187–203 (2002)
8. Gelder, A.V., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
9. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Kowalski, R., Bowen, K. (eds.) 5th Conference on Logic Programming, pp. 1070–1080. MIT Press, Cambridge (1988)
10. Kakas, A.C., Kowalski, R.A., Toni, F.: The role of abduction in logic programming. In: Gabbay, D., Hogger, C.J., Robinson, J.A. (eds.) Handbook in Artificial Intelligence and Logic Programming, vol. 5, pp. 235–324. Oxford University Press, Oxford (1998)
11. Kakas, A.C., Mancarella, P.: Generalized stable models: A semantics for abduction. In: ECAI, pp. 385–391 (1990)
12. Nieves, J.C.: Modeling arguments and uncertain information — A non-monotonic reasoning approach. PhD thesis, Software Department (LSI), Technical University of Catalonia (2008)
13. Nieves, J.C., Osorio, M.: A General Schema For Generating Argumentation Semantics From Logic Programming Semantics. Research Report LSI-08-32-R, Technical University of Catalonia, Software Department, LSI (2008), http://www.lsi.upc.edu/dept/techreps/buscar.php
14. Nieves, J.C., Osorio, M., Cortés, U.: Preferred Extensions as Stable Models. Theory and Practice of Logic Programming 8(4), 527–543 (2008)
15. Nieves, J.C., Osorio, M., Cortés, U., Olmos, I., Gonzalez, J.A.: Defining new argumentation-based semantics by minimal models. In: Seventh Mexican International Conference on Computer Science (ENC 2006), pp. 210–220. IEEE Computer Society Press, Los Alamitos (2006)
16. Osorio, M., Navarro, J.A., Arrazola, J.R., Borja, V.: Logics with Common Weak Completions. Journal of Logic and Computation 16(6), 867–890 (2006)
17. Prakken, H., Vreeswijk, G.A.W.: Logics for defeasible argumentation. In: Gabbay, D., Günthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 4, pp. 219–318. Kluwer Academic Publishers, Dordrecht (2002)

Appendix A: Proof of Theorem 1

Proof. The proof is by construction.

First of all, we recall some definitions about the notion of generalized S model.

Let S be a logic programming semantics, P be a logic program and A be a set of atoms (called abductives) such that $A \subseteq \mathcal{L}_P$. We say that M_B is a *generalized*

S *model*[1] of P with respect to A if $M \in S(P \cup B)$ where $B \subseteq A$ and $M \subseteq \mathcal{L}_P$. It is also possible to define a partial order between generalized S models (with respect to A) of a program according to the set inclusion with respect to the subindex B. We say that M is a *minimal generalized* S *model* of P with respect to A if there exists a set of atoms B, such that M_B is a generalized S model of P with respect to A and M_B is minimal with respect to the partial order just defined.

We write S^* to denote the *minimal generalized* S *semantics*, where $A = \mathcal{L}_P$. Namely $S^*(P)$ is the collection of minimal generalized S models of P with respect to $\mathcal{L}_P$. Observe that in our definition we are not instantiating the definition to a particular logic programming semantics.

It is immediate to verify that for every semantics S and program P, $S^*(P)$ is defined.

Now, let S be a semantics that is always defined. We define the associate S^r semantics recursively as follow: Given a program P of order 0, $S^r(P) = S(P)$. For a program P of order $n > 0$ we define $S^r(P) = \bigcup_{M \in S(P_0)} \{M\} * S^r(R(Q, A))$ where $Q = P \setminus P_0$ and $A = \langle M; \mathcal{L}_{P_0} \setminus M \rangle$.

Note that if S is always defined then S^r is always defined. More over S^r is relevant by construction.

Our final semantics is the following: Let P be a normal program. Let $freetaut(P)$ be program P after removing every tautology. Let $tautp(P) = freetaut(P) \cup \{x \leftarrow x : x \in \mathcal{L}_P\}$. Then

$$S'(P) = S^{*^r}(tautp(P)).$$

Clearly $S'(P)$ is always defined and relevant and invariant under adding tautologies.

Appendix B: Proof of Theorem 2

In order to present the proof of Theorem 2, we are going to present some definitions *w.r.t.* the argumentation semantics CF2[2], and we are going to show some lemmas.

Definition 5. *A set S of arguments is said to be conflict-free if there are no arguments a, b in S such that a attacks b.*

We will denote by $max\_conflict\_freeSets(AF)$ the set of maximal conflict free sets (*w.r.t.* set inclusion) of an argumentation framework AF.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, the binary relation of *path-equivalence* between nodes, denoted as $PE_{AF} \subseteq (AR \times AR)$, is defined as follows:

— $\forall \alpha \in AR$, $(\alpha, \alpha) \in PE_{AF}$,

[1] The concept of generalized S model is closely related to the semantics of *abductive logic programming* [11,10], in particular to the concept of *generalized answer set*.

[2] The details of these definitions are presented in [2].

— given two distinct nodes α, $\beta \in AR$, $(\alpha, \beta) \in PE_{AF}$ if and only if there is a path from α to β and a path from β to α.

Given an argumentation framework $AF = \langle AR, attacks \rangle$, the *strongly connected components* of AF are the equivalence classes of nodes under the relation of path-equivalence. The set of the strongly connected components of AF is denoted as $SCCS_{AF}$. Given a node $\alpha \in AR$, the strongly connected component α belongs to is denoted as $SCC_{AF}(\alpha)$.

Now, given an argumentation framework, let $AF = \langle AR, attacks \rangle$, and $S \subseteq AR$, the restriction of AF to S is the argumentation framework $AF{\downarrow_S}{=}\langle S, attacks \cap (S \times S)\rangle$.

Considering an argumentation framework, $AF = \langle AR, attacks \rangle$, a set $E \subseteq AR$ and a strongly connected component $S \in SCCS_{AF}$, the set $D_{AF}(S, E)$ consists of the nodes of S attacked by E from outside S, the set $U_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are defended by E (i.e., their defeaters from outside S are all attacked by E), and $P_{AF}(S, E)$ consists of the nodes of S that are not attacked by E from outside S and are not defended by E (i.e., at least one of their defeaters from outside S is not attacked by E). Finally, $UP_{AF}(S, E) = (S \setminus D_{AF}(S, E)) = (U_{AF}(S, E) \cup P_{AF}(S, E))$.

Here, we define $GF(AF, C)$ for an argumentation framework $AF{=}\langle AR, attacks \rangle$ and a set $C \subseteq A$, representing the defended nodes of AF: two cases have to be considered in this respect.

If AF consists of exactly one strongly connected component, it does not admit a decomposition where to apply the directionality principle, therefore it has to be assumed that $GF(AF, C)$ coincides in this case with a *base function*, denoted as $BF_S(AF, C)$, that must be assigned in order to characterize a particular argumentation semantics S.

On the other hand, if AF can be decomposed into several strongly connected components, then, $GF(AF, C)$ is obtained by applying recursively GF to each strongly connected component of AF, deprived of the nodes in $D_{AF}(S, E)$. Formally, this means that for any $S \in SCCS_{AF}$, $(E \cap S) \in GF(AF \downarrow_{UP_{AF}(S,E)}, C')$, where C' represents the set of defended nodes of the restricted argumentation framework $AF \downarrow_{UP_{AF}(S,E)}$. The set C' can be determined taking into account both the attacks coming from outside AF and those coming from other strongly connected components of AF.

Definition 6. *A given argumentation semantics S is* SCC-recursive *if and only if for any argumentation framework $AF{=}\langle AR, \text{attacks}\rangle$, $E_S(AF){=}GF(AF, AR)$, where for any $AF = \langle AR, \text{attacks}\rangle$ and for any set $C \subseteq AR$, the function $GF(AF, C) \subseteq 2^{AR}$ is defined as follows: for any $E \subseteq AR$, $E \in GF(AF, C)$ if and only if*

- *in case $|SCCS_{AF}| = 1$, $E \in BF_S(AF, C)$,*
- *otherwise, $\forall S \in SCCS_{AF} (E \cap S) \in GF(AF \downarrow_{UP_{AF(S,E)}}, U_{AF}(S, E) \cap C)$.*

where $BF_S(AF, C)$ is a function, called base function, that, given an argumentation framework $AF = \langle AR, \text{attacks}\rangle$ such that $|SCCS_{AF}| = 1$ and a set $C \subseteq AR$, gives a subset of 2^{AR}.

Observe that Definition 6 does not define any particular semantics, essentially it defines a general schema for defining argumentation semantics. In particular, when $BF_S(AF, C)$ is instantiated by the function which returns the maximal conflict free sets of AF *w.r.t.* C, Definition 6 defines $CF2$.

The following lemma shows that the number of strongly connected components of an argumentation framework AF is the same to the number of components of the normal logic program P_{AF}.

Lemma 2. *Let AF be an argumentation framework. If $P_{AF} = P^1_{AF} \cup P^2_{AF}$ such that P^1_{AF} is of order n, then $|SCC_{AF}| = n + 1$.*

Proof. (sketch) Since the number of components of P^1_{AF} depends on the number of equivalent classes of atoms in $\mathcal{L}_{P_{AF}}$ and the number of strongly connected components depends on the number of equivalent classes of nodes in PE_{AF}, the proof follows from that fact that:

- The number of equivalent classes of atoms induced by the relation *depends on* in $\mathcal{L}_{P_{AF}}$ is the same to the number of classes of nodes induces by the relation *path-equivalence* in PE_{AF}.

Lemma 3. *Let $AF = \langle AR, Attacks\rangle$ be an argumentation framework. If $P_{AF} = P^1_{AF} \cup P^2_{AF}$ such that P^1_{AF} is of order 0, then $E \in max\_conflict\_freeSets(AF)$ if and only if $\{a(a)|a \in E\} \cup \{d(a)|a \in AR \setminus E\}$ is a minimal model of P_{AF}.*

Proof. Observations:

1. M is a minimal model of P_{AF} if and only if there exists M_1 and M_2 such that $M = M_1 \cup M_2$, M_1 is a minimal model of P^1_{AF} and $M_2 = \{a(a)|a(a) \leftarrow \neg d(a) \in P^2_{AF}, d(a) \notin M_1\}$.
2. If E is a conflict free set of AF, then $M = \{d(a)|a \in AR \setminus E\}$ is a model of P^1_{AF}.
3. If M is a model of P^1_{AF}, then $E = \{a|a(a) \in M\}$ is a conflict free set of AF.

=> If E is a maximal conflict free set of AF, then, by Proposition 1 of [14] and Observation 2, $M_1 = \{d(a)|a \in AR \setminus E\}$ is a minimal model of P^1_{AF}. Hence, by Observation 1, $M_1 \cup \{a(a)|a \in E\}$ is a minimal model of P_{AF}.

<= If $M = M_1 \cup M_2$ such that $E \subseteq AR$, $M_1 = \{d(a)|a \in AR \setminus E\}$, $M_2 = \{a(a)|a \in E\}$ and M is a minimal model of P_{AF}, hence by Observation 1, M_1 is a minimal model of P^1_{AF}. Therefore, by Observation 3 and Proposition 1 of [14], E is a maximal conflict free of AF.

Given the set of strongly connected components $SCC(AF)$, we denote by $\leq_{SCC}$ the partial order between strongly connected components defined in [2]. This partial order is induced by the so called directionality principle and the relation of attack between set of arguments.

Main Proof

Proof. **Theorem 2.** (sketch) Since the construction of both semantics is recursive, the proof is by induction *w.r.t.* the number of components n of the normal logic program P_{AF}.

Base Step. If $n = 0$, then AF has just one strongly connected component (Lemma 2); hence, $MM^r_{Arg}(AF) = CF2(AF)$ by Lemma 3.

Inductive Step. If $n > 0$, then the proof follows from the following observations:

1. The partial order $\leq_{SCC}$ and the partial order $<_P$ define equivalent classes of sets of arguments of AF and atoms in $\mathcal{L}_{P_{AF}}$ respectively.
2. The base function for the construction of MM^r_{Arg} and $CF2(AF)$ are equivalent (Lemma 3).

Appendix C: Proof of Lemma 1

Proof. **Lemma 1.** (sketch)
The proof follows from the following observations:

1. By the definition of the stratified argumentation semantics, the decision of knowing if the set of arguments E is a stratified extension of AF is reduced to the decision problem of knowing if a given set of atoms M is a minimal model of a logic program P (it is a consequence of the base case of the recursive function $MM^r_c(P)$, see Definition 2).
2. Since there is a relationship between minimal models and logic consequence (see Lemma 1 of [14]), the decision problem of knowing if $M \subseteq \mathcal{L}_{P_{AF}}$ is a minimal model of P_{AF} can be reduced to the decision problem of 2-UNSAT.
3. It is known that the decision problem of 2-UNSAT is polynomial time computable [1].

Deep Inference in Bi-intuitionistic Logic

Linda Postniece

Logic and Computation Group
College of Computer Science and Engineering
The Australian National University
Linda.Postniece@anu.edu.au

Abstract. Bi-intuitionistic logic is the extension of intuitionistic logic with exclusion, a connective dual to implication. Cut-elimination in bi-intuitionistic logic is complicated due to the interaction between these two connectives, and various extended sequent calculi, including a display calculus, have been proposed to address this problem.

In this paper, we present a new extended sequent calculus DBiInt for bi-intuitionistic logic which uses nested sequents and "deep inference", i.e., inference rules can be applied at any level in the nested sequent. We show that DBiInt can simulate our previous "shallow" sequent calculus LBiInt. In particular, we show that deep inference can simulate the residuation rules in the display-like shallow calculus LBiInt. We also consider proof search and give a simple restriction of DBiInt which allows terminating proof search. Thus our work is another step towards addressing the broader problem of proof search in display logic.

1 Introduction

Bi-intuitionistic logic (BiInt) is the extension of intuitionistic logic with exclusion $-\!\!<$ (also known as "subtraction" and "co-implication"), a connective dual to implication $\rightarrow$. In a sequent calculus setting, the left-introduction rule for exclusion is dual to the right introduction rule for implication:

$$\frac{A \Rightarrow B, \Delta}{A -\!\!< B \Rightarrow \Delta} -\!\!<_L \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \rightarrow B} \rightarrow_R$$

BiInt was first studied by Rauszer as a Hilbert calculus with algebraic and Kripke semantics [13]. More recently, Crolard has investigated applications of BiInt to type theory [3]. The duality between implication and exclusion also makes it interesting to study BiInt purely from a proof-theoretic point of view, since cut-elimination in BiInt is non-trivial. That is, the only cut-free calculi for BiInt either use extended sequent mechanisms such as labels [12], variables [6] or nested sequents [8], or display calculi that rely on residuation [5].

In this paper we follow up on our previous work on nested sequent calculi for BiInt [8], as well as our more recent work on deep inference for tense logics [7]. Nested sequents are structures that can be seen as trees of traditional sequents, and have been studied, among others, by Kashima [11] and Brünnler [2] in the

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 320–334, 2009.

context of classical modal and tense logics. Nested sequent calculi allow either "shallow" or "deep" inference: in shallow inference calculi inference rules are applied at the top level only, and residuation rules are used to re-orient the trees to bring the required structures to the top-level. In deep inference calculi, inference rules can be applied at any level, and propagation rules move formulae around the trees. We are not aware of any nested sequent calculi for intuitionistic logics that use deep inference, although there has been work on deep inference in the calculus of structures for intuitionistic logic [14]; see Section 6 for details.

In [7], we showed that deep inference in nested sequents for tense logic can simulate the residuation rules of a shallow inference nested sequent calculus. Since BiInt is the intuitionistic analog of tense logic, a natural question is whether deep inference can be applied to BiInt. We address this question here, and show that indeed we can simulate residuation using deep inference for BiInt.

More precisely, we show that residuation, an operation on the trees encoded in nested sequents, can be simulated by propagation rules, which only move formulae between the nodes rather than change the shape of the trees. Due to the intuitionistic nature of the logic, some differences from [7] arise. Namely, we introduce the concept of polarity, and some rules are only applicable in positive or negative sub-structures, rather than general sub-structures as in tense logic.

The rest of the paper is organised as follows. First, in Section 2, we give the syntax of BiInt as well as our nested sequent structures. We then present two nested sequent calculi for BiInt in Section 3: we recall the shallow inference calculus **LBiInt** [8] and give a new deep inference calculus **DBiInt**. In Section 4, we show that provability in **DBiInt** is equivalent to provability in **LBiInt**, which is the central result of our paper. The non-trivial part is showing that the residuation rules of **LBiInt** can be simulated by the propagation rules and deep inference of **DBiInt**. In Section 5, we give a simple restriction of **DBiInt** that allows terminating backward proof search. In Section 6, we describe related work and outline future work. The Appendix contains more detailed proofs.

2 Nested Sequents

The **formulae** of BiInt are built from a set of atoms *Atoms* according to the following grammar, where $p \in \mathit{Atoms}$:

$$A := p \mid \top \mid \bot \mid A \to A \mid A \mathrel{-\!\!<} A \mid A \wedge A \mid A \vee A.$$

A structure is defined by the following grammar, where A is a BiInt formula:

$$X := \emptyset \mid A \mid (X, X) \mid X \triangleright X.$$

The structural connective "," (comma) is associative and commutative and $\emptyset$ is its unit. We always consider structures modulo these equivalences. To reduce parentheses, we assume that "," binds tighter than "$\triangleright$". Thus, we write $X, Y \triangleright Z$ to mean $(X, Y) \triangleright Z$. If X and Y are structures, then $X \triangleright Y$ is a **nested deep sequent**, and $X \Rightarrow Y$ is a **nested shallow sequent**.

A **context** is a deep sequent with a single hole, and is written $\Sigma[]$. We write $\Sigma[X]$ to denote the sequent that results from filling the hole in $\Sigma[]$ with X. For example, if $\Sigma_1[] = Z_1 \rhd [], Z_2$ then $\Sigma_1[X_1 \rhd Y_1] = Z_1 \rhd (X_1 \rhd Y_1), Z_2$.

A hole in a context can have either negative or positive **polarity**. If $X, [] \rhd Y$ is a substructure of Σ, then $\Sigma[]$ is a negative context and we write $\Sigma^-[]$. If $X \rhd [], Y$ is a substructure of Σ, then $\Sigma[]$ is a positive context and we write $\Sigma^+[]$. For example, $\Sigma^-[] = X_1 \rhd ([] \rhd Y_1)$ is a negative context and $\Sigma^+[] = (X_1 \rhd []) \rhd Y_1$ is a positive context.

We define the **immediate super-structure** of a context as: $\overbrace{\Sigma[]} = X \rhd Y$ such that $X \rhd Y$ is a sub-structure of Σ and $X = [], X'$ for some structure X' or $Y = [], Y'$ for some structure Y'. We define the **top-level** formulae of a structure as: $\{X\} = \{A \mid X = (A, Y)$ for some A and $Y\}$. For example, if $\Sigma[] = A, B \rhd C, (D, (E \rhd F) \rhd [])$, then $\overbrace{\Sigma[G]} = (D, (E \rhd F) \rhd G)$, and $\{D, (E \rhd F)\} = \{D\}$.

While deep inference allows us to "zoom-in" to any sub-structure deep inside the nested sequent, the concept of an immediate super-structure acts the opposite way in that it allows us to "zoom-out" from a context to its immediate surrounding nested structure. This will be useful when we restrict our rules for terminating proof-search, allowing us to impose local checks on the rules.

3 Nested Sequent Calculi

We now present the two nested sequent calculi that we will use in the rest of the paper: the shallow inference calculus **LBiInt** from our previous work [8] and a new deep inference calculus **DBiInt**. We repeat the rules of **LBiInt** here to make the present paper self-contained, see [8] for the full details.

Fig. 1 gives the rules of the cut-free fragment of **LBiInt** (**LBiInt** has cut-elimination). Central to this calculus is the idea that inference rules can only be applied to formulae at the top level of nested sequents, and the structural rules s_L, s_R, $\rhd_L$ and $\rhd_R$ are used to bring the required sub-structures to the top level. These rules, very similar to residuation postulates in display logic, are essential for the cut-elimination proof of **LBiInt**, however, they contain too much non-determinism for effective proof search.

Note that we have changed the notation slightly from [8] for an expository purpose: we are using $\rhd$ as the only structural connective, while the original **LBiInt** had $<$ in negative sub-structures and $>$ in positive sub-structures. Also, the $\rhd_L$ and $\rhd_R$ rules contained an implicit weakening which we have removed here. We also use $\Rightarrow$ for the sequent turnstile, reserving $\vdash$ for denoting provability. That is, we write $\vdash_{\mathbf{LBiInt}} \Pi : X \Rightarrow Y$ to mean that there exists an **LBiInt**-derivation Π of the sequent $X \Rightarrow Y$.

Fig. 2 gives the rules of our new deep inference calculus **DBiInt**. Here the inference rules can be applied at any level of the nested sequent, indicated by the use of contexts. Notably, there are no residuation rules; indeed the main goal of our paper is to show that the residuation rules of **LBiInt** can be simulated by deep inference and propagation rules in **DBiInt**. We write $\vdash_{\mathbf{DBiInt}} \Pi : X \rhd Y$ to mean that there exists a **DBiInt**-derivation Π of the sequent $X \rhd Y$.

Identity and logical constants:

$$\frac{}{X, A \Rightarrow A, Y}\, id \qquad \frac{}{X, \bot \Rightarrow Y}\, \bot_L \qquad \frac{}{X \Rightarrow \top, Y}\, \top_R$$

Structural rules:

$$\frac{X \Rightarrow Y}{X, A \Rightarrow Y}\, w_L \qquad \frac{X \Rightarrow Y}{X \Rightarrow A, Y}\, w_R \qquad \frac{X, A, A \Rightarrow Y}{X, A \Rightarrow Y}\, c_L \qquad \frac{X \Rightarrow A, A, Y}{X \Rightarrow A, Y}\, c_R$$

$$\frac{(X_1 \triangleright Y_1), X_2 \Rightarrow Y_2}{X_1, X_2 \Rightarrow Y_1, Y_2}\, s_L \qquad \frac{X_1 \Rightarrow Y_1, (X_2 \triangleright Y_2)}{X_1, X_2 \Rightarrow Y_1, Y_2}\, s_R$$

$$\frac{X_2 \Rightarrow Y_2, Y_1}{(X_2 \triangleright Y_2) \Rightarrow Y_1}\, \triangleright_L \qquad \frac{X_1, X_2 \Rightarrow Y_2}{X_1 \Rightarrow (X_2 \triangleright Y_2)}\, \triangleright_R$$

Logical rules:

$$\frac{X, B_i \Rightarrow Y}{X, B_1 \wedge B_2 \Rightarrow Y}\, \wedge_L\ i \in \{1, 2\} \qquad \frac{X \Rightarrow A, Y \qquad X \Rightarrow B, Y}{X \Rightarrow A \wedge B, Y}\, \wedge_R$$

$$\frac{X, A \Rightarrow Y \qquad X, B \Rightarrow Y}{X, A \vee B \Rightarrow Y}\, \vee_L \qquad \frac{X \Rightarrow B_i, Y}{X \Rightarrow B_1 \vee B_2, Y}\, \vee_R\ i \in \{1, 2\}$$

$$\frac{X \Rightarrow A, Y \qquad X, B \Rightarrow Y}{X, A \to B \Rightarrow Y}\, \to_L \qquad \frac{X, A \Rightarrow B}{X \Rightarrow A \to B, Y}\, \to_R$$

$$\frac{A \Rightarrow B, Y}{X, A \prec B \Rightarrow Y}\, \prec_L \qquad \frac{X \Rightarrow A, Y \qquad X, B \Rightarrow Y}{X \Rightarrow A \prec B, Y}\, \prec_R$$

Fig. 1. LBiInt: a shallow inference system for BiInt

We write $|\Pi|$ for the height of a derivation, i.e., the number of sequents on the longest branch, where Π is either an **LBiInt**-derivation or a **DBiInt**-derivation.

3.1 Examples

We give two examples to illustrate the difference between shallow inference in **LBiInt** and deep inference in **DBiInt**.

Example 1. The following is a derivation of Uustalu's formula [12] in **LBiInt**:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \Rightarrow q, p}\, id \qquad \dfrac{}{p, q \Rightarrow q}\, id}{p \Rightarrow q, p \prec q}\, \prec_R}{p \triangleright q \Rightarrow p \prec q}\, \triangleright_L}{p \triangleright q, r \Rightarrow p \prec q}\, w_L \qquad \dfrac{}{(p \triangleright q), r \Rightarrow r}\, id}{(p \triangleright q), r \Rightarrow (p \prec q) \wedge r}\, \wedge_R}{p \triangleright q \Rightarrow r \to ((p \prec q) \wedge r)}\, \to_R}{p \Rightarrow q, r \to ((p \prec q) \wedge r)}\, s_L$$

This example uses the rules $\triangleright_L$ and s_L to bring the required sub-structures to the top-level to apply the inference rules.

Identity and logical constants:

$$\frac{}{\Sigma[X, A \triangleright A, Y]}\, id \qquad \frac{}{\Sigma^-[\bot]}\, \bot_L \qquad \frac{}{\Sigma^+[\top]}\, \top_R$$

Propagation rules:

$$\frac{\Sigma^-[\{X\}, (X \triangleright Y)]}{\Sigma^-[X \triangleright Y]}\, \triangleright_{L1} \qquad \frac{\Sigma^+[(X \triangleright Y), \{Y\}]}{\Sigma^+[X \triangleright Y]}\, \triangleright_{R1}$$

$$\frac{\Sigma[X \triangleright (W, (\{X\}, Y \triangleright Z))]}{\Sigma[X \triangleright (W, (Y \triangleright Z))]}\, \triangleright_{L2} \qquad \frac{\Sigma[((X \triangleright Y, \{Z\}), W) \triangleright Z]}{\Sigma[((X \triangleright Y), W) \triangleright Z]}\, \triangleright_{R2}$$

Logical rules:

$$\frac{\Sigma^-[A \vee B, A] \qquad \Sigma^-[A \vee B, B]}{\Sigma^-[A \vee B]}\, \vee_L \qquad \frac{\Sigma^+[A \vee B, A, B]}{\Sigma^+[A \vee B]}\, \vee_R$$

$$\frac{\Sigma^-[A \wedge B, A, B]}{\Sigma^-[A \wedge B]}\, \wedge_L \qquad \frac{\Sigma^+[A \wedge B, A] \qquad \Sigma^+[A \wedge B, B]}{\Sigma^+[A \wedge B]}\, \wedge_R$$

$$\frac{\Sigma^-[A \prec B, (A \triangleright B)]}{\Sigma^-[A \prec B]}\, \prec_L \qquad \frac{\Sigma^+[A \to B, (A \triangleright B)]}{\Sigma^+[A \to B]}\, \to_R$$

$$\frac{\Sigma[X, A \to B \triangleright A, Y] \qquad \Sigma[X, A \to B, B \triangleright Y]}{\Sigma[X, A \to B \triangleright Y]}\, \to_L$$

$$\frac{\Sigma[X \triangleright Y, A \prec B, A] \qquad \Sigma[X, B \triangleright Y, A \prec B]}{\Sigma[X \triangleright Y, A \prec B]}\, \prec_R$$

Fig. 2. DBiInt: a deep inference system for BiInt

Example 2. The following is a derivation of Uustalu's formula in **DBiInt** where we abbreviate $A = r \to ((p \prec q) \wedge r)$, $B = (p \prec q) \wedge r$ and $X = p, r \triangleright B, p \prec q$ to save space. For readability, we draw a box around the structure that the inference rule is applied to, unless it is the top-level structure:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \triangleright q, A, X, B, p \prec q, p}\, id \qquad \dfrac{}{p, q \triangleright q, A, X, B, p \prec q}\, id}{p \triangleright q, A, X, B, p \prec q}\, \prec_R}{p \triangleright q, A, \boxed{(p, r \triangleright B, p \prec q)}}\, \triangleright_{R1} \qquad \dfrac{}{p \triangleright q, A, \boxed{(p, r \triangleright B, r)}}\, id}{p \triangleright q, A, \boxed{(p, r \triangleright (p \prec q) \wedge r)}}\, \wedge_R}{p \triangleright q, A, (r \triangleright (p \prec q) \wedge r)}\, \triangleright_{L2}}{p \triangleright q, r \to ((p \prec q) \wedge r)}\, \to_R$$

This example uses deep inference to apply the inference rules at any level. The formula propagation rules $\triangleright_{R1}$ and $\triangleright_{L2}$ ensure that the required formulae are propagated to the appropriate sub-structure.

4 Soundness and Completeness of DBiInt

4.1 Soundness of DBiInt

We first show that the propagation rules of **DBiInt** can be derived in **LBiInt** using residuation. This is not a surprising result, since the residuation rules in display logics are used exactly for the purpose of displaying and un-displaying sub-sequents so that inference rules can be applied to them.

Theorem 1 (Soundness). *For any structures X and Y, if $\vdash_{\mathbf{DBiInt}} \Pi : X \triangleright Y$ then $\vdash_{\mathbf{LBiInt}} \Pi' : X \Rightarrow Y$.*

Proof. By induction on $|\Pi|$. We show one interesting case. The given **DBiInt**-derivation is on the left, and we obtain the **LBiInt**-derivation on the right, where Π'_1 is obtained by the induction hypothesis (IH).

$$\dfrac{\begin{array}{c}\Pi_1\\ X \triangleright (Y_1 \triangleright Y_2), \{Y_2\}\end{array}}{X \triangleright (Y_1 \triangleright Y_2)}\,{\triangleright}R1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\Pi'_1\\ X \Rightarrow (Y_1 \triangleright Y_2), \{Y_2\}\end{array}}{X, Y_1 \Rightarrow Y_2, \{Y_2\}}\,s_R}{X, Y_1 \Rightarrow Y_2}\,c_R}{X \Rightarrow (Y_1 \triangleright Y_2)}\,{\triangleright}R$$

4.2 Completeness of DBiInt

Our aim is to show that **DBiInt** is complete w.r.t. **LBiInt**. But first we state some basic lemmas, which can all be proved using simple induction on $|\Pi|$.

Lemma 1 (Admissibility of general weakening). *For any structures X and Y: if $\vdash_{\mathbf{DBiInt}} \Pi : \Sigma[X]$ then $\vdash_{\mathbf{DBiInt}} \Pi' : \Sigma[X, Y]$ such that $|\Pi'| \leq |\Pi|$.*

Lemma 2 (Invertibility). *All* **DBiInt** *rules are invertible: if the conclusion is derivable, so are all the premises.*

Lemma 3 (Admissibility of formula contraction). *For any structure X and formula A: if $\vdash_{\mathbf{DBiInt}} \Pi : \Sigma[X, A, A]$ then $\vdash_{\mathbf{DBiInt}} \Pi' : \Sigma[X, A]$.*

Corollary 1. *For any structure X, if $\vdash_{\mathbf{DBiInt}} \Pi : \Sigma[X, \{X\}]$ then $\vdash_{\mathbf{DBiInt}} \Pi' : \Sigma[X]$.*

We now show that the residuation rules of **LBiInt** are admissible in **DBiInt**; that is, they can be simulated by the propagation rules of **DBiInt**. Lemmas 4 to 7 are proved by induction on $|\Pi|$. We show some interesting cases, where Π ends with a propagation rule.

Lemma 4 (Admissibility of s_L). *If $\vdash_{\mathbf{DBiInt}} \Pi : (X \triangleright Y), Z \triangleright W$ then $\vdash_{\mathbf{DBiInt}} \Pi' : X, Z \triangleright Y, W$.*

Proof. Suppose Π ends as below left. Then we obtain a derivation Π_1' of $X, \{X\}, Z \triangleright Y, W$ from the IH, and a derivation Π_1'' of $X, Z \triangleright Y, W$ from Corollary 1 and Π_1'. Then the derivation on the right gives the required:

$$\dfrac{\begin{array}{c}\Pi_1\\ \{X\}, (X \triangleright Y), Z \triangleright W\end{array}}{(X \triangleright Y), Z \triangleright W}\,{\triangleright_{L1}} \quad \rightsquigarrow \quad \begin{array}{c}\Pi_1''\\ X, Z \triangleright Y, W\end{array}$$

Lemma 5 (Admissibility of s_R). *If* $\vdash_{\mathbf{DBiInt}} \Pi : X \triangleright Y, (Z \triangleright W)$ *then* $\vdash_{\mathbf{DBiInt}} \Pi' : X, Z \triangleright Y, W$.

Lemma 6 (Admissibility of $\triangleright_L$). *If* $\vdash_{\mathbf{DBiInt}} \Pi : X \triangleright Y, Z$ *then* $\vdash_{\mathbf{DBiInt}} \Pi' : (X \triangleright Y) \triangleright Z$.

Lemma 7 (Admissibility of $\triangleright_R$). *If* $\vdash_{\mathbf{DBiInt}} \Pi : X, Y \triangleright Z$ *then* $\vdash_{\mathbf{DBiInt}} \Pi' : X \triangleright (Y \triangleright Z)$.

Proof. Suppose Π ends as below left. Then we obtain a derivation Π_1' of $(X_1 \triangleright X_2, \{Z\}) \triangleright (Y \triangleright Z)$ from the IH, and a derivation Π_1'' of $(X_1 \triangleright X_2, \{Z\}) \triangleright ((Y \triangleright Z), \{Z\})$ from Lemma 1 and Π_1'. Then the derivation on the right gives the required:

$$\dfrac{\begin{array}{c}\Pi_1\\ (X_1 \triangleright X_2, \{Z\}), Y \triangleright Z\end{array}}{(X_1 \triangleright X_2), Y \triangleright Z}\,{\triangleright_{R2}} \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{array}{c}\Pi_1''\\ (X_1 \triangleright X_2, \{Z\}) \triangleright ((Y \triangleright Z), \{Z\})\end{array}}{(X_1 \triangleright X_2) \triangleright ((Y \triangleright Z), \{Z\})}\,{\triangleright_{R2}}}{(X_1 \triangleright X_2) \triangleright (Y \triangleright Z)}\,{\triangleright_{R1}}$$

Lemma 8 (Admissibility of general contraction). *For any structures X and Y: if* $\vdash_{\mathbf{DBiInt}} \Pi : \Sigma[X, Y, Y]$ *then* $\vdash_{\mathbf{DBiInt}} \Pi' : \Sigma[X, Y]$.

Proof. By induction on the size of Y. The interesting case is when $Y = (Y_1 \triangleright Y_2)$. This can be reduced to contractions on Y_1 and Y_2, which are admissible by the IH; $Y_1 \triangleright Y_2$ can then be reconstructed using Lemmas 4 to 7.

Theorem 2 (Completeness). *For any structures X and Y, if* $\vdash_{\mathbf{LBiInt}} \Pi : X \Rightarrow Y$ *then* $\vdash_{\mathbf{DBiInt}} \Pi' : X \triangleright Y$.

Proof. By induction on $|\Pi|$. We illustrate one case where Π ends in a logical rule application and one where Π ends in a structural rule application. The other interesting cases use Lemmas 4 to 7.

– Suppose Π is as below left. Then we first obtain **DBiInt**-derivations Π_1' and Π_2' of $X \triangleright A, Y$ and $X, B \triangleright Y$ respectively from the IH. Second, we obtain Π_1'' and Π_2'' by Lemma 1 from Π_1' and Π_2'. Finally, the required **DBiInt**-derivation is as below right:

$$\dfrac{\begin{array}{c}\Pi_1\\ X \Rightarrow A, Y\end{array} \quad \begin{array}{c}\Pi_2\\ X, B \Rightarrow Y\end{array}}{X, A \to B \Rightarrow Y}\,{\to_L} \quad \rightsquigarrow \quad \dfrac{\begin{array}{c}\Pi_1''\\ X, A \to B \triangleright A, Y\end{array} \quad \begin{array}{c}\Pi_2''\\ X, A \to B, B \triangleright Y\end{array}}{X, A \to B \triangleright Y}\,{\to_L}$$

– Suppose Π is as below left. Then we first obtain a **DBiInt**-derivation Π_1' of $X, Y \triangleright Z$ by the IH. Second, we obtain a **DBiInt**-derivation Π_1'' of $X \triangleright (Y \triangleright Z)$ from Π_1' by Lemma 7. Then the required **DBiInt**-derivation is as below right:

$$\dfrac{\begin{array}{c}\Pi_1\\ X, Y \Rightarrow Z\end{array}}{X \Rightarrow (Y \triangleright Z)}\,{\triangleright_R} \qquad \rightsquigarrow \qquad \begin{array}{c}\Pi_1''\\ X \triangleright (Y \triangleright Z)\end{array}$$

Theorem 3. *For any structures X and Y, $\vdash_{\mathbf{LBiInt}} \Pi : X \Rightarrow Y$ if and only if $\vdash_{\mathbf{DBiInt}} \Pi' : X \triangleright Y$.*

Proof. By Theorems 1 and 2.

5 Proof Search

Naive proof search in **DBiInt** does not terminate. Consider the following proof attempt fragment, where $X = (A \to B) \to C, (D \to E) \to F$ and we only show the left premise of each $\to_L$ rule instance:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{X \triangleright G, A \to B, (X, A \triangleright B, D \to E, (X, A, D \triangleright E, A \to B, (A \triangleright B)))}\,{\to_R}}{X \triangleright G, A \to B, (X, A \triangleright B, D \to E, \boxed{(X, A, D \triangleright E, A \to B)})}\,{\to_L}}{X \triangleright G, A \to B, \boxed{(X, A \triangleright B, D \to E, (X, A, D \triangleright E))}}\,{\triangleright_{L2}}}{X \triangleright G, A \to B, \boxed{(X, A \triangleright B, D \to E, (D \triangleright E))}}\,{\to_R}}{X \triangleright G, A \to B, \boxed{(X, A \triangleright B, D \to E)}}\,{\to_L}}{X \triangleright G, A \to B, \boxed{(X, A \triangleright B)}}\,{\triangleright_{L2}}}{X \triangleright G, A \to B, (A \triangleright B)}\,{\to_R}}{X \triangleright G, A \to B}\,{\to_L}}{(A \to B) \to C, (D \to E) \to F \triangleright G}\,{\to_L}$$

There is an interaction between the $\to_R$, $\triangleright_{L2}$ and $\to_L$ rules that causes non-termination, even for the intuitionistic fragment of the logic. This well-known problem occurs in traditional sequent calculi as well, and it is caused by the implicit contraction in the $\to_L$ rule. For intuitionistic logic, this problem has been addressed by contraction-free calculi [4] and history-based loop-checks [10]. However, these methods are less suitable for BiInt where the interaction between $\to$ and $-\!\!<$ formulae needs to be considered. Here we address termination using a saturation process and two derived rules that speed up proof search. The approach is similar to our previous work [8], but here we apply it to deep inference and contexts instead of top-level sequents only.

Let $-\!\!<_{L1}$ and $\to_{R1}$ denote two rules derived as below, where a dashed inference line means the conclusion is derived from the premise using Lemma 1:

$$\dfrac{\dfrac{\dfrac{\Sigma^-[A, A{-\!\!\prec}B]}{\Sigma^-[A{-\!\!\prec}B, A, (A \triangleright B)]}\ \text{Lemma 1}}{\Sigma^-[A{-\!\!\prec}B, (A \triangleright B)]}\ \triangleright_{L1}}{\Sigma^-[A{-\!\!\prec}B]}\ {-\!\!\prec}_L \quad \rightsquigarrow \quad \dfrac{\Sigma^-[A, A{-\!\!\prec}B]}{\Sigma^-[A{-\!\!\prec}B]}\ {-\!\!\prec}_{L1}$$

$$\dfrac{\dfrac{\dfrac{\Sigma^+[A \to B, B]}{\Sigma^+[A \to B, (A \triangleright B), B]}\ \text{Lemma 1}}{\Sigma^+[A \to B, (A \triangleright B)]}\ \triangleright_{R1}}{\Sigma^+[A \to B]}\ \to_R \quad \rightsquigarrow \quad \dfrac{\Sigma^+[A \to B, B]}{\Sigma^+[A \to B]}\ \to_{R1}$$

Let $\Sigma[Z]$ be any sequent. Then let $X \triangleright Y = \overbrace{\Sigma[Z]}$. We say that $\Sigma[Z]$ is **saturated** iff all the following conditions are met:

1. $\{X\} \cap \{Y\} = \emptyset$
2. If $A \wedge B \in \{X\}$ then $A \in \{X\}$ and $B \in \{X\}$
3. If $A \wedge B \in \{Y\}$ then $A \in \{Y\}$ or $B \in \{Y\}$
4. If $A \vee B \in \{X\}$ then $A \in \{X\}$ or $B \in \{X\}$
5. If $A \vee B \in \{Y\}$ then $A \in \{Y\}$ and $B \in \{Y\}$
6. If $A \to B \in \{X\}$ then $A \in \{Y\}$ or $B \in \{X\}$
7. If $A{-\!\!\prec}B \in \{Y\}$ then $A \in \{Y\}$ or $B \in \{X\}$
8. If $A \to B \in \{Y\}$ then $B \in \{Y\}$
9. If $A{-\!\!\prec}B \in \{X\}$ then $A \in \{X\}$

Let X and Y be two structures. We say that a formula $A \to B$ is **realised** by $X \triangleright Y$ iff there exists a structure $Z \triangleright W \in Y$ such that $A \in Z$ and $B \in W$. We say that a formula $C{-\!\!\prec}D$ is **realised** by $X \triangleright Y$ iff there exists a structure $Z \triangleright W \in X$ such that $C \in Z$ and $D \in W$. We define the super-set relation on sequents as follows: $X_1 \triangleright Y_1 \supset X_0 \triangleright Y_0$ iff $\{X_1\} \supset \{X_0\}$ or $\{Y_1\} \supset \{Y_0\}$. Then the following simple modifications of **DBiInt** ensure termination using only local checks:

Definition 1. *Let* $\mathbf{DBiInt_1}$ *be the system obtained from* **DBiInt** *with the following changes:*

1. *Add the derived rules* ${-\!\!\prec}_{L1}$ *and* $\to_{R1}$.
2. *Replace rules* ${-\!\!\prec}_L$, $\to_R$ *by the following:*

$$\dfrac{\Sigma^-[A{-\!\!\prec}B, (A \triangleright B)]}{\Sigma^-[A{-\!\!\prec}B]}\ {-\!\!\prec}_L$$

where $\Sigma^-[A{-\!\!\prec}B]$ *is saturated and* $A{-\!\!\prec}B$ *is not realised by* $\overbrace{\Sigma^-[A{-\!\!\prec}B]}$

$$\dfrac{\Sigma^+[A \to B, (A \triangleright B)]}{\Sigma^+[A \to B]}\ \to_R$$

where $\Sigma^+[A \to B]$ *is saturated and* $A \to B$ *is not realised by* $\overbrace{\Sigma^+[A \to B]}$

3. *Replace rules* $\triangleright_{L2}$ *and* $\triangleright_{R2}$ *by the following:*

$$\frac{\Sigma[X \triangleright (W, (\{X\}, Y \triangleright Z))]}{\Sigma[X \triangleright (W, (Y \triangleright Z))]} \triangleright_{L2} \quad \textit{where } \{X\} \supset \{Y\}$$

$$\frac{\Sigma[((X \triangleright Y, \{Z\}), W) \triangleright Z]}{\Sigma[((X \triangleright Y), W) \triangleright Z]} \triangleright_{R2} \quad \textit{where } \{Z\} \supset \{Y\}$$

4. *Replace rules* $\rightarrow_L$, $-\!\!<_R$, $\triangleright_{L1}$, $\triangleright_{R1}$, $\wedge_L$, $\wedge_R$, $\vee_L$, $\vee_R$ *with the following restricted versions:*
   (a) *Let* γ_0 *be the conclusion of the rule let* γ_1 *(and* γ_2*) be the premises. The rule is applicable only if:* $\overbrace{\gamma_1} \supset \overbrace{\gamma_0}$ *and* $\overbrace{\gamma_2} \supset \overbrace{\gamma_0}$.

Theorem 4. *For any structures* X *and* Y, $\vdash_{\mathbf{DBiInt}} \Pi : X \triangleright Y$ *if and only if* $\vdash_{\mathbf{DBiInt_1}} \Pi' : X \triangleright Y$.

Theorem 5. *For any* X *and* Y, *backward proof search in* **DBiInt$_1$** *for* $X \triangleright Y$ *terminates.*

6 Related Work, Future Work and Conclusion

Deep inference: Deep inference in the calculus of structures was pioneered by Guglielmi [9]. In his work, inference rules can be applied deep inside formulae, not just deep inside nested sequent structures as in our case. This method has also been applied to intuitionistic logic [14]. The works of Kashima [11] and Brünnler [2] are closer to ours since their deep inference rules are applied to nested structures (Brünnler calls them deep sequents). However, both [11] and [2] only cover classical modal and tense logics, while we have extended the notion of deep inference to bi-intuitionistic logic using polarised contexts.

Taming display logic: Areces and Bernardi [1] appear to be the first to have noticed the connection between deep inference and residuation in display logic in the context of categorial grammar. However, they do not give an explicit proof of this correspondence as we have done here for our calculi.

Extensions and restrictions: Since BiInt is a conservative extension of intuitionistic logic, our calculi are also sound and complete for the intuitionistic fragment of BiInt: we simply need to ignore all rules for $-\!\!<$. We are also interested in extending our technique to similar logics such as Lambek-Grishin logic. Since many of our proofs use associativity and commutativity, it is not obvious that our technique will be immediately applicable to substructural logics.

Our contributions: The main contribution of our paper is showing that deep inference in nested sequent calculi for bi-intuitionistic logic can mimic residuation in display-like calculi. Thus our work is another step towards addressing the broader problem of proof search in display logic. Secondly, our calculus **DBiInt** and its restriction **DBiInt$_1$** are interesting calculi for proof search in BiInt in their own right. We leave the details of an efficient implementation of **DBiInt$_1$** for future work.

Acknowledgements. We would like to thank Rajeev Goré, Alwen Tiu and the anonymous reviewers for their comments on an earlier version of this paper.

References

1. Areces, C., Bernardi, R.: Analyzing the core of categorial grammar. Journal of Logic, Language, and Information 13(2), 121–137 (2004)
2. Brünnler, K.: Deep sequent systems for modal logic. In: Governatori, G., et al. (eds.) Advances in Modal Logic, vol. 6, pp. 107–119. College Publications (2006)
3. Crolard, T.: A formulae-as-types interpretation of Subtractive Logic. Journal of Logic and Computation 14(4), 529–570 (2004)
4. Dyckhoff, R.: Contraction-free sequent calculi for intuitionistic logic. The Journal of Symbolic Logic 57(3), 795–807 (1992)
5. Goré, R.: Substructural logics on display. LJIGPL 6(3), 451–504 (1998)
6. Goré, R., Postniece, L.: Combining derivations and refutations for cut-free completeness in bi-intuitionistic logic. Journal of Logic and Computation. To appear, Advance Access, http://logcom.oxfordjournals.org/cgi/content/abstract/exn067
7. Goré, R., Postniece, L., Tiu, A.: Taming displayed tense logics using nested sequents with deep inference. To appear in Proceedings of TABLEAUX 2009 (2009)
8. Goré, R., Postniece, L., Tiu, A.: Cut-elimination and proof-search for bi-intuitionistic logic using nested sequents. In: Advances in Modal Logic, vol. 7, pp. 43–66. College Publications (2008)
9. Guglielmi, A.: A system of interaction and structure. ACM Trans. Comput. Log. 8(1) (2007)
10. Heuerding, A., Seyfried, M., Zimmermann, H.: Efficient loop-check for backward proof search in some non-classical propositional logics. In: Miglioli, P., Moscato, U., Ornaghi, M., Mundici, D. (eds.) TABLEAUX 1996. LNCS, vol. 1071, pp. 210–225. Springer, Heidelberg (1996)
11. Kashima, R.: Cut-free sequent calculi for some tense logics. Studia Logica 53, 119–135 (1994)
12. Pinto, L., Uustalu, T.: Proof search and counter-model construction for bi-intuitionistic propositional logic with labelled sequents. To appear in Proceedings of TABLEAUX 2009 (2009)
13. Rauszer, C.: An algebraic and Kripke-style approach to a certain extension of intuitionistic logic. Dissertationes Mathematicae 168 (1980)
14. Tiu, A.: A local system for intuitionistic logic. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS, vol. 4246, pp. 242–256. Springer, Heidelberg (2006)

A Proofs

Proof of Theorem 1:

Proof. By induction on $|\Pi|$. We show the interesting cases. In each case the given **DBiInt**-derivation is on the left, and we obtain the **LBiInt**-derivation on the right, where Π_1' is obtained by the IH.

$$\dfrac{\begin{array}{c}\Pi_1\\ \{X_1\}, (X_1 \triangleright X_2) \triangleright Y\end{array}}{(X_1 \triangleright X_2) \triangleright Y}\,{\triangleright}L1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\Pi_1'\\ \{X_1\}, (X_1 \triangleright X_2) \Rightarrow Y\end{array}}{\{X_1\}, X_1 \Rightarrow X_2, Y}\,s_L}{X_1 \Rightarrow X_2, Y}\,c_L}{(X_1 \triangleright X_2) \Rightarrow Y}\,{\triangleright}_L$$

$$\dfrac{\dfrac{\Pi_1}{X \triangleright (W, (\{X\}, Y_1 \triangleright Y_2))}}{X \triangleright (W, (Y_1 \triangleright Y_2))}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Pi_1'}{X \Rightarrow W, (\{X\}, Y_1 \triangleright Y_2)}}{X \triangleright W \Rightarrow \{X\}, Y_1 \triangleright Y_2}s_R}{(X \triangleright W), \{X\}, Y_1 \Rightarrow Y_2}\triangleright_L}{(X \triangleright W), \{X\} \Rightarrow Y_1 \triangleright Y_2}\triangleright_R}{X \triangleright W \Rightarrow \{X\} \triangleright (Y_1 \triangleright Y_2)}\triangleright_R}{X \Rightarrow W, (\{X\} \triangleright (Y_1 \triangleright Y_2))}s_L}{X, \{X\} \Rightarrow W, (Y_1 \triangleright Y_2)}s_R}{X \Rightarrow W, (Y_1 \triangleright Y_2)}c_L$$

$$\dfrac{\dfrac{\Pi_1}{((X \triangleright Y, \{Z\}), W) \triangleright Z}}{((X \triangleright Y), W) \triangleright Z}\triangleright_{R2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Pi_1'}{(X \triangleright Y, \{Z\}), W \Rightarrow Z}}{X \triangleright Y, \{Z\} \Rightarrow W \triangleright Z}\triangleright_R}{X \Rightarrow Y, \{Z\}, (W \triangleright Z)}s_L}{X \triangleright Y \Rightarrow \{Z\}, (W \triangleright Z)}\triangleright_L}{(X \triangleright Y) \triangleright \{Z\} \Rightarrow W \triangleright Z}\triangleright_L}{(X \triangleright Y) \triangleright \{Z\}, W \Rightarrow Z}s_R}{(X \triangleright Y), W \Rightarrow \{Z\}, Z}s_L}{(X \triangleright Y), W \Rightarrow Z}c_R$$

Each of the following proofs is by induction on $|\Pi|$. Π_1' is obtained from Π_1 using the IH. A dashed inference line labeled W means that the conclusion is obtained from the premise using Lemma 1. A dashed inference line labeled C means that the conclusion is obtained from the premise using Corollary 1. **Proof of Lemma 4:**

Proof.

$$\dfrac{\dfrac{\Pi_1}{\Sigma[(X \triangleright (\{X\}, Y_1 \triangleright Y_2)), Z \triangleright W]}}{\Sigma[(X \triangleright (Y_1 \triangleright Y_2)), Z \triangleright W]}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X, Z \triangleright (\{X\}, Y_1 \triangleright Y_2), W]}}{\Sigma[X, Z \triangleright (Y_1 \triangleright Y_2), W]}\triangleright_{L2} W$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[(X \triangleright Y), Z \triangleright (W_1 \triangleright W_2), \{W_2\}}}{\Sigma[(X \triangleright Y), Z \triangleright (W_1 \triangleright W_2)}\triangleright_{R1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X, Z \triangleright Y, (W_1 \triangleright W_2), \{W_2\}]}}{\Sigma[X, Z \triangleright Y, (W_1 \triangleright W_2)]}\triangleright_{R1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[(X \triangleright Y, \{W\}), Z \triangleright W]}}{\Sigma[(X \triangleright Y), Z \triangleright W]}\triangleright_{R2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X, Z \triangleright Y, \{W\}, W]}}{\Sigma[X, Z \triangleright Y, W]} C$$

Proof of Lemma 5:

Proof.

$$\dfrac{\dfrac{\Pi_1}{\Sigma[\{X_1\},(X_1 \triangleright X_2) \triangleright Y,(Z \triangleright W)]}}{\Sigma[(X_1 \triangleright X_2) \triangleright Y,(Z \triangleright W)]}\triangleright_{L1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[\{X_1\},(X_1 \triangleright X_2),Z \triangleright Y,W]}}{\Sigma[(X_1 \triangleright X_2),Z \triangleright Y,W]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright Y,(\{Z_1\},(Z_1 \triangleright Z_2) \triangleright W)]}}{\Sigma[X \triangleright Y,(Z_1 \triangleright Z_2 \triangleright W)]}\triangleright_{L1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X,\{Z_1\},(Z_1 \triangleright Z_2) \triangleright Y,W]}}{\Sigma[X,(Z_1 \triangleright Z_2) \triangleright Y,W]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright Y,(\{X\},Z \triangleright W)]}}{\Sigma[X \triangleright Y,(Z \triangleright W)]}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X,\{X\},Z \triangleright Y,W]}}{\Sigma[X,Z \triangleright Y,W]}\,C$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright Y,(Z \triangleright W),\{W\}]}}{\Sigma[X \triangleright Y,(Z \triangleright W)]}\triangleright_{R1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X,Z \triangleright Y,W,\{W\}]}}{\Sigma[X,Z \triangleright Y,W]}\,C$$

$$\dfrac{\dfrac{\Pi_1'}{\Sigma[X \triangleright Y,((Z_1 \triangleright Z_2,\{W\}) \triangleright W)]}}{\Sigma[X \triangleright Y,((Z_1 \triangleright Z_2) \triangleright W)]}\triangleright_{R2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X,(Z_1 \triangleright Z_2,\{W\}) \triangleright Y,W]}}{\Sigma[X,(Z_1 \triangleright Z_2) \triangleright Y,W]}\triangleright_{R2}$$

Proof of Lemma 6:

Proof.

$$\dfrac{\dfrac{\Pi_1}{\Sigma[\{X_1\},(X_1 \triangleright X_2) \triangleright Y,Z]}}{\Sigma[(X_1 \triangleright X_2) \triangleright Y,Z]}\triangleright_{L1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[(\{X_1\},(X_1 \triangleright X_2) \triangleright Y) \triangleright Z]}}{\Sigma[((X_1 \triangleright X_2) \triangleright Y) \triangleright Z]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright (\{X\},Y_1 \triangleright Y_2),Z]}}{\Sigma[X \triangleright (Y_1 \triangleright Y_2),Z]}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[(X \triangleright (\{X\},Y_1 \triangleright Y_2)) \triangleright Z]}}{\Sigma[(X \triangleright (Y_1 \triangleright Y_2)) \triangleright Z]}\triangleright_{L2}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright Y,(\{X\},Z_1 \triangleright Z_2)]}}{\Sigma[X \triangleright Y,(Z_1 \triangleright Z_2)]}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\Pi_1'}{\Sigma[(X \triangleright Y) \triangleright (\{X\},Z_1 \triangleright Z_2)]}}{\Sigma[\{X\},(X \triangleright Y) \triangleright (\{X\},Z_1 \triangleright Z_2)]}\,W}{\Sigma[\{X\},(X \triangleright Y) \triangleright (Z_1 \triangleright Z_2)]}\triangleright_{L2}}{\Sigma[(X \triangleright Y) \triangleright (Z_1 \triangleright Z_2)]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright (Y_1 \triangleright Y_2),\{Y_2\},Z]}}{\Sigma[X \triangleright (Y_1 \triangleright Y_2),Z]}\triangleright_{R1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[(X \triangleright (Y_1 \triangleright Y_2),\{Y_2\}) \triangleright Z]}}{\Sigma[(X \triangleright (Y_1 \triangleright Y_2)) \triangleright Z]}\triangleright_{R1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X \triangleright Y,(Z_1 \triangleright Z_2),\{Z_2\}]}}{\Sigma[X \triangleright Y,(Z_1 \triangleright Z_2)]}\triangleright_{R1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[(X \triangleright Y) \triangleright (Z_1 \triangleright Z_2),\{Z_2\}]}}{\Sigma[(X \triangleright Y) \triangleright (Z_1 \triangleright Z_2)]}\triangleright_{R1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[(X_1 \triangleright X_2, \{Y,Z\}) \triangleright Y, Z]}}{\Sigma[(X_1 \triangleright X_2) \triangleright Y, Z]}\triangleright_{R2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\Pi_1'}{\Sigma[((X_1 \triangleright X_2, \{Y,Z\}) \triangleright Y) \triangleright Z]}}{\Sigma[((X_1 \triangleright X_2, \{Y,Z\}) \triangleright Y, \{Z\}) \triangleright Z]}W}{\Sigma[((X_1 \triangleright X_2) \triangleright Y, \{Z\}) \triangleright Z]}\triangleright_{R2}}{\Sigma[((X_1 \triangleright X_2) \triangleright Y) \triangleright Z]}\triangleright_{R2}$$

Proof of Lemma 7:

Proof.

$$\dfrac{\dfrac{\Pi_1}{\Sigma[\{X_1\}, (X_1 \triangleright X_2), Y \triangleright Z]}}{\Sigma[(X_1 \triangleright X_2), Y \triangleright Z]}\triangleright_{L1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\Pi_1'}{\Sigma[(X_1 \triangleright X_2) \triangleright (\{X_1\}, Y \triangleright Z)]}}{\Sigma[(\{X_1\}, (X_1 \triangleright X_2)) \triangleright (\{X_1\}, Y \triangleright Z)]}W}{\Sigma[(\{X_1\}, (X_1 \triangleright X_2)) \triangleright (Y \triangleright Z)]}\triangleright_{L2}}{\Sigma[(X_1 \triangleright X_2) \triangleright (Y \triangleright Z)]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X, \{Y_1\}, (Y_1 \triangleright Y_2) \triangleright Z]}}{\Sigma[X, (Y_1 \triangleright Y_2) \triangleright Z]}\triangleright_{L1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X \triangleright (\{Y_1\}, (Y_1 \triangleright Y_2) \triangleright Z)]}}{\Sigma[X \triangleright ((Y_1 \triangleright Y_2) \triangleright Z)]}\triangleright_{L1}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X, Y \triangleright (\{X,Y\}, Z_1 \triangleright Z_2)]}}{\Sigma[X, Y \triangleright (Z_1 \triangleright Z_2)]}\triangleright_{L2} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\Pi_1'}{\Sigma[X \triangleright (Y \triangleright (\{X,Y\}, Z_1 \triangleright Z_2))]}}{\Sigma[X \triangleright (\{X\}, Y \triangleright (\{X,Y\}, Z_1 \triangleright Z_2))]}W}{\Sigma[X \triangleright (\{X\}, Y \triangleright (Z_1 \triangleright Z_2))]}\triangleright_{L2}}{\Sigma[X \triangleright (Y \triangleright (Z_1 \triangleright Z_2))]}\triangleright_{L2}$$

$$\dfrac{\dfrac{\Pi_1}{\Sigma[X, Y \triangleright (Z_1 \triangleright Z_2), \{Z_2\}]}}{\Sigma[X, Y \triangleright (Z_1 \triangleright Z_2)]}\triangleright_{R1} \quad \rightsquigarrow \quad \dfrac{\dfrac{\Pi_1'}{\Sigma[X \triangleright (Y \triangleright ((Z_1 \triangleright Z_2), \{Z_2\}))]}}{\Sigma[X \triangleright (Y \triangleright (Z_1 \triangleright Z_2))]}\triangleright_{R1}$$

Proof of Lemma 8:

Proof. By induction on the size of Y. The interesting case is when $Y = (Y_1 \triangleright Y_2)$. We show how this can be reduced to contractions on Y_1 and Y_2, which are admissible by the IH. A dashed inference line means that the conclusion is obtained from the premise using the respective Lemma or the IH. Suppose we have Y in a negative context, the other case is symmetric:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Sigma[(Y_1 \triangleright Y_2), (Y_1 \triangleright Y_2) \triangleright Z]}{\Sigma[Y_1, (Y_1 \triangleright Y_2) \triangleright Y_2, Z]}\text{ Lemma 4}}{\Sigma[Y_1, Y_1 \triangleright Y_2, Y_2, Z]}\text{ Lemma 4}}{\Sigma[Y_1 \triangleright Y_2, Y_2, Z]}\text{ IH}}{\Sigma[Y_1 \triangleright Y_2, Z]}\text{ IH}}{\Sigma[(Y_1 \triangleright Y_2) \triangleright Z]}\text{ Lemma 6}$$

Proof of Theorem 4:

Proof. For the left-to-right direction, use induction on $|\Pi|$. The interesting cases are when $|\Pi|$ ends with rule instance that does not meet one of the restrictions 2 to 4 imposed by Definition 1.

- Suppose restriction 2 of the rule $\rightarrow_R$ is not met, so that $\Sigma^+[A \rightarrow B]$ is not saturated. Then we use Lemma 2 and the IH to permute the offending rule instance upwards.
- Suppose restriction 2 of the rule $\rightarrow_R$ is not met, so that $A \rightarrow B$ is in fact realised by $\overbrace{\Sigma^+[A \rightarrow B]}$. Then Π is as below:

$$\dfrac{\begin{array}{c}\Pi_1 \\ \Sigma^+[A \rightarrow B, (A \triangleright B), (X, A \triangleright B, Y)]\end{array}}{\Sigma^+[A \rightarrow B, (X, A \triangleright B, Y)]} \rightarrow_R$$

  Then by the IH, there exists a $\mathbf{DBiInt_1}$-derivation Π_2 of $\Sigma^+[A \rightarrow B, (A \triangleright B), (X, A \triangleright B, Y)]$. By Lemma 1, there exists a $\mathbf{DBiInt_1}$-derivation Π_3 of $\Sigma^+[A \rightarrow B, (X, A \triangleright B, Y), (X, A \triangleright B, Y)]$. Finally, by Lemma 8, there exists a $\mathbf{DBiInt_1}$-derivation Π_1'' of $\Sigma^+[A \rightarrow B, (X, A \triangleright B, Y)]$.

The right-to-left direction is obvious, since every rule of $\mathbf{DBiInt_1}$ is a rule of **DBiInt**, or can be derived in **DBiInt**.

Proof sketch of Theorem 5:

We can define a translation from $\mathbf{DBiInt_1}$ sequents to trees, similar to the one in [7]. We then show that the depth of the trees is bounded (using restriction 2 of Definition 1) and that the size of the nodes is bounded (using restrictions 3 and 4 of Definition 1).

$\mathcal{CL}$: An Action-Based Logic for Reasoning about Contracts*

Cristian Prisacariu and Gerardo Schneider

Department of Informatics, University of Oslo,
P.O. Box 1080 Blindern, N-0316 Oslo, Norway
{cristi,gerardo}@ifi.uio.no

Abstract. This paper presents a new version of the $\mathcal{CL}$ contract specification language. $\mathcal{CL}$ combines deontic logic with propositional dynamic logic but it applies the modalities exclusively over structured actions. $\mathcal{CL}$ features synchronous actions, conflict relation, and an action negation operation. The $\mathcal{CL}$ version that we present here is more expressive and has a cleaner semantics than its predecessor. We give a direct semantics for $\mathcal{CL}$ in terms of *normative structures*. We show that $\mathcal{CL}$ respects several desired properties from legal contracts and is decidable. We relate this semantics with a trace semantics of $\mathcal{CL}$ which we used for run-time monitoring contracts.

1 Introduction

Internet-based negotiation and contracting is becoming more diverse and complex in the e-business and e-government environments. This calls for a more formal apparatus which can be used by the computer to automate and help in some of the tasks involved in e-contracting; e.g. detection of contradictions and inconsistencies in contracts, identification of superfluous clauses, and checking some desired properties on a contract. This contracting style found in e-business and virtual organizations (and inspired from legal contracts) can also be used in service oriented architectures, component based systems [1], and agent societies [2]. In these areas contracts are used to regulate the interaction and exchanges between the parties involved (being that services, components, or agents).

Much research has been invested into giving a formalization of contractual clauses, and also into providing a machine readable language for specifying contracts. Such a formal language is desired for doing static (like model-checking) or dynamic (like run-time monitoring) analysis of (abstractions of) contracts. Moreover, the automation of the negotiation process becomes a feasible goal. The most promising approaches are based on variants of deontic logic.

In this paper we present a logic (which we call $\mathcal{CL}$) designed to represent and reason about contracts. The goal of $\mathcal{CL}$ is to preserve many of the natural properties and concepts relevant to legal contracts, while avoiding deontic paradoxes, and

* Partially supported by the Nordunet3 project "COSoDIS – Contract-Oriented Software Development for Internet Services" (http://www.ifi.uio.no/cosodis/).

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 335–349, 2009.

at the same time to have a suitable language for the specification of software contracts. $\mathcal{CL}$ combines deontic logic [3] with propositional dynamic logic (PDL) [4,5]. $\mathcal{CL}$ applies the (deontic and dynamic) modalities *exclusively* over actions instead of over formulas (or state of affairs) as is the case in standard deontic logic (SDL) [3]. Therefore, $\mathcal{CL}$ adopts what is known as the ought-to-do approach to deontic logic as opposed to the ought-to-be approach of SDL. The ought-to-do approach has been advocated by von Wright [6] which argued that deontic logic would benefit from a "foundation of actions", and many of the philosophical paradoxes of SDL would be eliminated. Important contributions to this approach were done by Segerberg to introduce actions inside the deontic modalities [7] and by the seminal work of Meyer on dynamic deontic logic (DDL) [8,9].

$\mathcal{CL}$ extends the (regular) actions, which are usually considered in DDL and PDL, with a concurrency operator to model the notion of actions "*done at the same time*". The model of concurrency that we adopt is the synchrony model of Milner's SCCS [10]. Synchrony is easy to integrate with the other regular operations on actions (choice, sequence, and repetition). Moreover, synchrony is a natural choice for reasoning about the notion "at the same time" for human-like actions that we have in legal contracts.

The notion of synchrony has different meanings in different areas of computer science. Here we take the distinction between *synchrony* and *asynchrony* as presented in the SCCS calculus and later implemented in e.g. the Esterel synchronous programming language [11]. We understand *asynchrony* as when two concurrent systems proceed at indeterminate relative speeds (i.e. their actions may have different non-correlated durations); whereas in the *synchrony* model each of the two concurrent systems instantaneously perform a single action at each time instant. This is an abstract view of the actions found in contracts and is good for reasoning about properties of the contract.

The *synchrony model* of concurrency takes the assumption that time is discrete and that basic actions are instantaneous and represent the time step. Moreover, at each time step all possible actions are performed, i.e. the system is considered *eager* and *active*. For this reason, if at a time point an obligation to perform an action is enabled, then this action must be immediately executed so that the obligation is not violated. The mathematical framework of the synchrony model is much cleaner and more general than the asynchronous interleaving model (SCCS has the (asynchronous) CCS as a subcalculus [10]).

Because of the assumption of an eager behavior for the actions the scope of the obligations (and of the other deontic modalities too) is immediate, making them *transient* obligations which are enforced only in the current world. One can get persistent obligations by using temporal operators, like the *always* operator ($\mathcal{CL}$ can encode temporal operators with the dynamic modality [5]). The eagerness assumption facilitates reasoning both about the existence of the deontic modalities as well as about violations of the obligations or prohibitions.

Initial investigations into $\mathcal{CL}$ have been done in [12]. The $\mathcal{CL}$ language presented in this paper is more expressive. A variant of this syntax of the $\mathcal{CL}$ (without the propositional constants) was used in [13] for doing run-time monitoring

of electronic contracts. There we used a restricted semantics based on traces of actions. This semantics was specially designed for monitoring the actions of the contracting parties at run-time with the purpose of detecting when a contract is violated. In [12] there was no direct semantics for $\mathcal{CL}$. In the present paper we give full semantics for $\mathcal{CL}$ based on *normative structures* and relate it with the trace based semantics of [13].

In this paper we present theoretical results and thus we use a simple and unrestricted syntax for $\mathcal{CL}$, in practice some syntactic restrictions may be imposed (see [12]). Since our main objective is to analyze contracts through formal tools like model-checking and run-time monitoring, we aim at a tractable language (i.e. decidable and with manageable complexities).

Related work: Compared to [8,9,14,15], which also consider deontic modalities applied over actions, the investigation presented in this paper at the level of the deontic actions is different in several ways. First we add a synchrony operation, and second, we exclude the Kleene star *. None of the few papers that consider repetition as an action combinator under deontic modalities [14,9] give a precise motivation for having such recurring actions inside obligations, permissions, or prohibitions. Even more, the use of repetition inside the deontic modalities is counter intuitive: take the expression $O(a^*)$ - "One is obliged to not pay, or pay once, or pay twice in a row, or..." - which puts no actual obligations; or take $P(a^*)$ - "One has the right to do any sequence of action a." - which is a very shallow permission and is captured by the widespread *Closure Principle* in jurisprudence where *what is not forbidden is permitted* [7]. Moreover, [9] argues that expressions like $F(a^*)$ and $P(a^*)$ should be simulated with the PDL modalities as $\langle a^*\rangle F(a)$ respectively $[a^*]P(a)$. In our opinion the * combinator under deontic modalities can be captured by using temporal or dynamic logic modalities along with deontic modalities over *-free actions.

Regarding the synchronous actions inside the dynamic modality, $\mathcal{CL}$ is included in the class of extensions of PDL which can reason about concurrent actions: PDL$^\cap$ with intersection of actions [16] which is undecidable for deterministic structures; or concurrent PDL [17]. In contrast, $\mathcal{CL}$ with synchronous composition over deterministic actions (see Definition 3) inside the dynamic modality is decidable. This makes $\mathcal{CL}$ more attractive for automation of reasoning about contracts.

Due to lack of space we do not present full proofs here; please refer to the technical reports [19,22] for proofs and more technical details.

2 The Contract Language $\mathcal{CL}$

The syntax of $\mathcal{CL}$ is defined by the grammar in Table 1. In what follows we provide intuitions of the $\mathcal{CL}$ syntax and define our notation and terminology.

We call a formula $\mathcal{C}$ a (general) *contract clause* (or plainly *contract*). We consider a finite number of propositional constants ϕ drawn from a set Φ_B. We call $O_{\mathcal{C}}(\alpha)$, $P(\alpha)$, and $F_{\mathcal{C}}(\alpha)$ the *deontic modalities*, representing the obligation, permission, or prohibition of performing a given *action* α. $\mathcal{CL}$ includes directly

Table 1. Syntax of the contract language $\mathcal{CL}$

$$\begin{array}{rcl} \mathcal{C} & := & \phi \mid O_{\mathcal{C}}(\alpha) \mid P(\alpha) \mid F_{\mathcal{C}}(\alpha) \mid \mathcal{C} \to \mathcal{C} \mid [\beta]\mathcal{C} \mid \ \bot \\ \alpha & := & a \mid \mathbf{0} \mid \mathbf{1} \mid \alpha \times \alpha \mid \alpha \cdot \alpha \mid \alpha + \alpha \\ \beta & := & a \mid \mathbf{0} \mid \mathbf{1} \mid \beta \times \beta \mid \beta \cdot \beta \mid \beta + \beta \mid \beta^* \mid \varphi? \\ \varphi? & := & \phi \mid \mathbf{0} \mid \mathbf{1} \mid \varphi? \vee \varphi? \mid \varphi? \wedge \varphi? \mid \neg\varphi? \end{array}$$

in the definition of the obligation and prohibition the *reparations* in case of violations. Intuitively $O_{\mathcal{C}}(\alpha)$ states the obligation to perform α, and the reparation $\mathcal{C}$ in case the obligation is *violated*, i.e. whenever α is not performed. The reparation may be any contract clause. The modality $O_{\mathcal{C}}(\alpha)$ (resp. $F_{\mathcal{C}}(\alpha)$) represents what is called contrary-to-duty obligations, CTDs, (resp. contrary-to-prohibitions, CTPs) in dynamic deontic logic.[1] Obligations without reparations are written as $O_{\bot}(\alpha)$ where $\bot$ (and conversely $\top$) is the Boolean `false` (respectively `true`). We usually write $O(\alpha)$ instead of $O_{\bot}(\alpha)$. The prohibition modality $F_{\mathcal{C}}(\alpha)$ states the actual forbidding of the action α together with the reparation $\mathcal{C}$ in case the prohibition is violated. Note that it is possible to express nested CTDs and CTPs. Permissions have no reparations associated because they cannot be violated; permissions can only be exercised.

Throughout the paper we denote by $a, b, c \in \mathcal{A}_B$ the *basic actions* (e.g. "pay", "deliver", or "redraw"), by indexed $\alpha \in \mathcal{A}$ *deontic actions*, and by indexed β the *dynamic actions*. Actions α are used inside the deontic modalities, whereas the (more general) actions β are used inside the dynamic modality. An *action* term α is constructed from the basic actions $a \in \mathcal{A}_B$ and the special actions $\mathbf{0}$ and $\mathbf{1}$ (called the *violating* action and respectively the *skip* action) using the binary constructors: *choice* "$+$", *sequence* "$\cdot$", and *synchrony* (or concurrency) "$\times$". Actions β have the extra operators Kleene star * (for *repetition*) and Boolean *test* $?$. Tests $\varphi?$ are constructed with the Boolean operators from *basic tests* which in our case are the propositional constants $\phi \in \Phi_B$ (also denoted $\mathcal{A}_B^?$)

We define a symmetric and irreflexive relation over the basic actions $\mathcal{A}_B$, which we call *conflict relation* and denote by $\#_{\mathcal{C}} \subseteq \mathcal{A}_B \times \mathcal{A}_B$. The conflict relation is a notion often found in legal contracts and is given a priori. The intuition of the conflict relation is that if two actions are in conflict then the actions cannot be done at the same time. This intuition explains the need for the following equational implication at the level of the deontic actions: $a \#_{\mathcal{C}} b \to a \times b = \mathbf{0}$, $\forall a, b \in \mathcal{A}_B$. This is necessary for detecting (and for ruling out) a first kind of *conflicts* in contracts: "Obligatory to go west and obligatory to go east" should result in a conflict (see Proposition 2-([16])). The second kind of conflicts that $\mathcal{CL}$ rules out are: "Obligatory to go west and forbidden to go west" which is a standard requirement on a deontic logic.

[1] The notions of CTD and CTP from $\mathcal{CL}$ are in contrast with the classical notion of CTD as found in the SDL literature [18]. In SDL, what we call reparations are secondary obligations which hold in the same world as the primary obligation. In our setting where the action changes the context (the world) one can see a violation of an obligation (or prohibition) only after the action is performed and thus the reparations are enforced in the changed context (next world).

The dynamic logic modality $[\cdot]\mathcal{C}$ is parameterized by *actions* β. The expression $[\beta]\mathcal{C}$ is read as: "after the action β is performed $\mathcal{C}$ must hold". Therefore, $\mathcal{CL}$ can reason about synchronous actions inside the dynamic modality. We use the classical Boolean implication operator $\rightarrow$; the other operators $\wedge, \vee, \neg, \leftrightarrow, \top, \oplus$ (exclusive or) are expressed in terms of $\rightarrow$ and $\bot$ as in propositional logic.

In $\mathcal{CL}$ we can write *conditional* obligations, permissions and prohibitions of two different kinds. As an example consider conditional obligations. The first kind is given with the propositional implication: $\mathcal{C} \rightarrow O_{\mathcal{C}}(\alpha)$ which is read as "if $\mathcal{C}$ is the case then it is obligatory that action α" (e.g. "If Internet traffic is high then the Client is obliged to pay"). The second kind is given with the dynamic box modality: $[\beta]O_{\mathcal{C}}(\alpha)$ which is read as "if action β was performed then it is obligatory that action α" (e.g. "after receiving necessary data the Provider is obliged to offer password").

The formalization of the actions has been thoroughly investigated in [19] where interpretations for the actions have been defined, and completeness and decidability results have been established. The semantics of the $\mathcal{CL}$ language is based on the interpretation of the deontic actions as rooted trees with edges labeled by elements of $2^{\mathcal{A}_B}$. Denote by $I(\alpha)$ the tree interpreting the deontic action α. Intuitively, $+$ provides the branching in the tree and $\cdot$ provides the parent-child relation on each branch (see also Theorem 3 in the appendix).

Each dynamic action β denotes a set of *guarded concurrent strings*.

Definition 1 (guarded concurrent strings). *Over the set of basic tests* $\mathcal{A}_B^?$ *we define* atoms *as functions* $\nu : \mathcal{A}_B^? \rightarrow \{0,1\}$ *which assign a Boolean value to each basic test. Denote by* $Atoms = \{0,1\}^{\mathcal{A}_B^?}$ *the set of all such functions. A* guarded concurrent string *(denoted by* u, v, w*) is a sequence*

$$w = \nu_0 x_1 \nu_1 \ldots x_n \nu_n, \qquad n \geq 0,$$

where $\nu_i \in Atoms$ *and* $x_i \in 2^{\mathcal{A}_B}$ *are sets of basic actions.*

For each dynamic action β we can construct a special two level finite automaton (denoted $GNFA(\beta)$) which accepts all and only the guarded concurrent strings denoting β. The important detail is that with each state of the automaton of the upper level it is associated a special finite state automaton (denoted $\lceil s \rceil$) which accepts a set of atoms. (An atom ν can be seen as a valuation of a test $\varphi?$ iff the truth assignments of ν to the basic tests make $\varphi?$ true; thus, for each test $\varphi?$ there is a set of all atoms which make it true.) The results of [19] ensure that working with the dynamic actions or with the automata on guarded concurrent strings is the same (they are different notations for the same set).

Proposition 1 (automata for tests). *There exists a class of finite state automata, denoted* $\mathcal{M}$*, which accept all and only the subsets of Atoms; in notation* $\mathcal{L}(M) \in 2^{Atoms}$, $M \in \mathcal{M}$ *and* $\forall A \in 2^{Atoms}, \exists M \in \mathcal{M}$ *s.t.* $\mathcal{L}(M) = A$.

Definition 2 (automata on guarded concurrent strings). *Consider a two level finite automaton* $GNFA = (S, \mathcal{P}(\mathcal{A}_B), S_0, \rho, F, \lceil \cdot \rceil)$*. It consists at the first level of a finite automaton on concurrent strings* $(S, \mathcal{P}(\mathcal{A}_B), S_0, \rho, F)$*, together*

with a map $\lceil\cdot\rceil : S \to \mathcal{M}$*. An automaton on concurrent strings (i.e. the first level automation) consists of a finite set of* states S*, the finite* alphabet $2^{\mathcal{A}_B}$ *(i.e. the powerset of the set of basic actions* $\mathcal{A}_B$*), a set of* initial *states* $S_0 \subseteq S$*, a* transition relation $\rho : 2^{\mathcal{A}_B} \to S \times S$*, and a set of final states* F*. At the lower level the mapping* $\lceil\cdot\rceil$ *associates with each state of the first level an automaton* $M \in \mathcal{M}$ *as defined in Proposition 1 which accepts a set of atoms denoted* $\mathcal{L}(\lceil s\rceil)$*.*

Intuitively, a *GNFA*(β) accepts a guarded concurrent string $\nu_0 x_1 \nu_1 \dots x_n \nu_n$ if there is a sequence of nodes $s_0 \dots s_n \in S$ with $s_n \in F$ s.t. the x_i label the transitions s_{i-1}, s_i and the atoms ν_i are accepted by the corresponding automata (on the second level) $\lceil s_i\rceil$. The definition with two levels of *GNFA* is needed when defining the special operations for fusion product and synchronous composition corresponding to respectively $\cdot$ and $\times$ (see [19]).

3 Semantics

The formulas $\mathcal{C}$ of the logic are given a model theoretic semantics in terms of (what we define as) *normative structures*.

Definition 3 (normative structure). *A* normative structure *is a tuple denoted* $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ *where* $\mathcal{W}$ *is a set of* worlds, $\mathcal{V} : \mathcal{W} \to 2^{\Phi_B}$ *is a* valuation function *returning for each world the set of propositional constants which hold in that world.* $\mathcal{A}_B$ *is a finite set of* basic labels *and* $2^{\mathcal{A}_B}$ *represents the* labels *of the structure as sets of basic labels;* $\rho : 2^{\mathcal{A}_B} \to 2^{\mathcal{W}\times\mathcal{W}}$ *is a function returning for each label a partial function (therefore for each label from one world there is at most one reachable world), and* $\varrho : \mathcal{W} \to 2^{\Psi}$ *is a marking function which marks each state with one or several markers from* $\Psi = \{\circ_a, \bullet_a \mid a \in \mathcal{A}_B\}$*. The marking function respects the restriction that no state can be marked by both* $\circ_a$ *and* $\bullet_a$ *(for any* $a \in \mathcal{A}_B$*). A* pointed normative structure *is a normative structure with a designated state* i *(denoted by* $K^{\mathcal{N}}, i$*).*

Notation: We denote by t a node of a tree (or by r the root) and by s (or i for initial) a world of a normative structure. Henceforth we use the more graphical notation $t \xrightarrow{\alpha_\times} t'$ ($s \xrightarrow{\beta_\times} s'$) for an edge (transition) in a tree (normative structure), where $\alpha_\times, \beta_\times \in 2^{\mathcal{A}_B}$ denote labels. Note that we consider both the trees and the normative structures to have the same set of basic labels $\mathcal{A}_B$. For the sake of notation we can view the valuation of one particular world $\mathcal{V}(s)$ as a function from Φ_B to $\{0, 1\}$ where $\forall\varphi \in \Phi_B$, $\mathcal{V}(s)(\varphi) = 1$ iff $\varphi \in \mathcal{V}(s)$ and 0 otherwise. Therefore, we can say that $\mathcal{V}(s)$ is accepted by the automata of the second level of *GNFA* and write $\mathcal{V}(s) \in \mathcal{L}(\lceil s\rceil)$.

One difference from the standard PDL is that we consider *deterministic* structures. This is natural and desired in legal contracts as opposed to the programming languages community where nondeterminism is an important notion. In contracts the outcome of an action like "deposit 100$ in the bank account" is uniquely determined. The deterministic restriction of Kripke structures was investigated in [20]. Note that deterministic PDL is undecidable if action negation (or intersection of actions) is added [5].

The marking function and the markers are needed to identify obligatory (i.e. $\circ$) and prohibited (i.e. $\bullet$) actions. Markers with different purposes were used in [8] to identify violations of obligations, in [14] to mark permitted transitions, and in [15] to identify permitted events. The labels of the normative structure (and of the trees $I(\alpha)$ or automata $GNFA(\beta)$) are sets of basic labels $\mathcal{A}_B$. Therefore, we can compare them using set inclusion (and call one label *bigger* than another).

Definition 4 (simulation). *For a tree $T = (\mathcal{N}, \mathcal{E}, \mathcal{A}_B)$ and a normative structure $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ we define a relation $\mathcal{S} \subseteq \mathcal{N} \times \mathcal{W}$ which we call the* simulation *of the tree node by the state of the structure.*

$t \,\mathcal{S}\, s$ *iff* $\forall t \xrightarrow{\gamma} t' \in T,\ \exists s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ *s.t.* $\gamma \subseteq \gamma' \wedge t' \mathcal{S} s'$ *and*

$$\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}} \text{ with } \gamma \subseteq \gamma' \text{ then } t' \,\mathcal{S}\, s'.$$

We say that a tree T, with root r is simulated by a normative structure $K^{\mathcal{N}}$ w.r.t. a state s, denoted $T \,\mathcal{S}_s\, K^{\mathcal{N}}$, iff $r \,\mathcal{S}\, s$.

Note two differences with the classical definition of simulation: first, the labels of the normative structure may be bigger than the labels in the tree because respecting an obligatory action means executing an action which includes it (is bigger). We can drop this condition and consider only $\gamma = \gamma'$, in which case we call the relation *strong simulation* and denote by $\mathcal{S}^s$. Second, any transition in the normative structure that can simulate an edge in the tree must enter under the simulation relation. This is because from the state s' onwards we need to be able to continue to look in the structure for the remaining tree (to see that it is simulated). We can weaken the definition by combining this condition with the one before into: $\forall t \xrightarrow{\gamma} t' \in T, \forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ with $\gamma \subseteq \gamma'$ then $t' \,\tilde{\mathcal{S}}\, s'$. We call the resulting relation *partial simulation* and denote it by $\tilde{\mathcal{S}}$.

Definition 5 (semantics). *We give in Table 2 a recursive definition of the* satisfaction relation $\models$ *of a formula $\mathcal{C}$ w.r.t. a pointed normative structure $K^{\mathcal{N}}, i$; it is written $K^{\mathcal{N}}, i \models \mathcal{C}$ and is read as "$\mathcal{C}$ is* satisfied *in the normative structure $K^{\mathcal{N}}$ at state i". The notions of* satisfiability *and* validity *are defined as usual.*

$\mathcal{CL}$ has particular properties found in legal contracts (which we list in Propositions 2 and 3). Some intuitive motivation for these properties from the perspective of legal contracts has been given in [12]. Here we explain how these properties influenced our decisions in the design of the semantics of $\mathcal{CL}$.

For the $O_{\mathcal{C}}$ the semantics has basically two parts. The first part (lines one to four) gives the interpretation of the obligation. Line one says how to walk on the structure depending on the tree of the action α. The normative structure must simulate the tree of the action, completely. This is because all the actions (i.e. on all the choices) that are obligatory must appear as transitions in the structure in order to guarantee properties like (21) and (8). Moreover, the simulation relation allowes for the labels of the structure to be bigger than (not necessary the same as) the labels in the tree. This is to guarantee property (7). Intuitively it means that if we do these (bigger) actions the obligation of α is still respected. The second condition of the simulation relation is needed also for properties like (8)

Table 2. Semantics for $\mathcal{CL}$

$K^{\mathcal{N}}, i \models \varphi$ iff $\varphi \in \mathcal{V}(i)$.

$K^{\mathcal{N}}, i \models \mathcal{C}_1 \to \mathcal{C}_2$ iff whenever $K^{\mathcal{N}}, i \models \mathcal{C}_1$ then $K^{\mathcal{N}}, i \models \mathcal{C}_2$.

$K^{\mathcal{N}}, i \models O_{\mathcal{C}}(\alpha)$ iff $I(\alpha)\ \mathcal{S}_i\ K^{\mathcal{N}}$, and
$\forall t \xrightarrow{\gamma} t' \in I(\alpha), \forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ s.t. $t\,\mathcal{S}\,s \wedge \gamma \subseteq \gamma'$ then
$\forall a \in \gamma, \circ_a \in \varrho(s')$, and
$\forall s \xrightarrow{\gamma'} s' \in K^{I(\alpha),i}_{rem}, \forall a \in \gamma'$ then $\circ_a \notin \varrho(s')$, and
$K^{\mathcal{N}}, s \models \mathcal{C}$ $\forall s \in N$ with $t\,\mathcal{S}^s\,s \wedge t \in \mathit{leaves}(I(\overline{\alpha}))$.

$K^{\mathcal{N}}, i \models F_{\mathcal{C}}(\alpha)$ iff $I(\alpha)\ \tilde{\mathcal{S}}_i\ K^{\mathcal{N}}$ then
$\forall \sigma \in I(\alpha)$ a full path s.t. $\sigma\ \mathcal{S}_i\ K^{\mathcal{N}}$,
$\exists t \xrightarrow{\gamma} t' \in \sigma$ s.t. $\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ with $t\,\mathcal{S}\,s \wedge \gamma \subseteq \gamma'$ then
$\forall a \in \gamma', \bullet_a \in \varrho(s')$, and
$K^{\mathcal{N}}, s \models \mathcal{C}$ $\forall s \in N$ with $t\,\mathcal{S}\,s \wedge t \in \mathit{leaves}(\sigma)$.

$K^{\mathcal{N}}, i \models P(\alpha)$ iff $I(\alpha)\ \mathcal{S}_i\ K^{\mathcal{N}}$, and
$\forall t \xrightarrow{\gamma} t' \in I(\alpha), \forall s \xrightarrow{\gamma} s' \in K^{\mathcal{N}}$ s.t. $t\,\mathcal{S}^s\,s \wedge t'\,\mathcal{S}^s\,s'$ then
$\forall a \in \gamma, \bullet_a \notin \varrho(s')$.

$K^{\mathcal{N}}, i \models [\beta]\mathcal{C}$ iff $\forall t \in \mathcal{W}$ with $(i,t) \in \rho(\beta)$ then $K^{\mathcal{N}}, t \models \mathcal{C}$.

$$\rho(\beta) = \{(s,t) \mid \exists k, \exists \sigma = x_0 \ldots x_k \text{ a final path in } \mathit{GNFA}(\beta), \exists s_0 \ldots s_k \in \mathcal{W} \text{ s.t. } s_0 = s, s_k = t, \text{ and } \forall i \le k,\ \mathcal{V}(s_i) \in \mathcal{L}(\lceil x_i \rceil), \text{ and } \forall 0 \le i < k \text{ with } x_i \xrightarrow{\beta_\times} x_{i+1} \in \sigma \text{ then } (s_i, s_{i+1}) \in \rho(\beta_\times)\}$$

because regardless of the way to respect a first obligation we must be able to enforce the subsequent obligations; which is related to property (8).

Lines two and three in the semantics of $O_{\mathcal{C}}$ say how the states must be marked with $\circ$. Note that the markers correspond exactly to the basic actions in the tree and not to the basic actions in the structure (which may be more). This is needed in the proof of the property (7) and property (17) and, in conjunction with the restriction on the normative structures, that no state is marked with both markers $\circ_a$ and $\bullet_a$ for any $a \in \mathcal{A}_B$, also for properties like (6) and (14). Moreover, note that all the relevant transitions in the normative structure must enter under the marking function in order to have properties like (18).

Line four ensures that no other reachable relevant transitions of the structure (i.e. from the non-simulating remainder structure $K^{I(\alpha),i}_{rem}$) are marked with obligation markers $\circ$. (See Definition 8 of $K^{I(\alpha),i}_{rem}$ in Appendix.) This is needed for property (22). It says that all the transitions which are outside the action (say outside α) should not be labeled with $\circ$ markers. Thus transitions from any other actions of a choice (e.g. of α' from $\alpha + \alpha'$) cannot be marked correctly.

The second part of the semantics of $O_{\mathcal{C}}$ is just the last line and handles our notion of reparation in case of violations of obligations. The negation of a compound action $\overline{\alpha}$ encodes all possible ways of violating the obligation $O(\alpha)$. Therefore, at the end of each of these possible ways of violation the reparation must be enforced. (See the Definition 9 of $\overline{\alpha}$ in Appendix.)

The conflict relation $\#_{\mathcal{C}}$ with the equational implication $a \#_{\mathcal{C}} b \rightarrow a \times b = \mathbf{0}$ and properties (1) and (7) is needed to prove how $\mathcal{CL}$ avoids conflicts like (16).

In the semantics of $F_{\mathcal{C}}$ the first condition uses partial simulation. This is because we want to capture the assumption that if a transition is not present in the normative structure then this is forbidden by default. In the second condition we consider all the paths in the tree which are simulated by the structure (i.e. come from the partial simulation) to have the property (10); prohibition of a choice must prohibit all. Note that we are interested only in *full* paths simulated by the structure because for the other paths some of the transitions are missing in the structure and thus there is some action on the sequence which is forbidden. On the other hand, in the third condition we consider at least one of the transitions on this full path in order to have the property (11); forbidding a sequence means forbidding some action on that sequence. Moreover, note that the $\bullet$ markers are associated with the labels in the normative structure and not with the labels in the tree (as for $O_{\mathcal{C}}$) in order to capture the property (9); i.e., forbidding an action implies forbidding any action that is bigger. Also note that all the transitions in the normative structure that simulate the chosen transition in the path are considered so that to capture the property (23). The last condition takes care to put the reparations where a violation of a prohibition is observed; i.e. after executing one full path in the tree of the prohibited action.

The semantics of P is similar to that of $O_{\mathcal{C}}$. It considers the simulation relation in order to have properties like (12). The difference is that the states *must not* be marked with $\bullet$ markers in order to have properties (15), (3), and (4); and to capture the principle that *what is not forbidden is permitted*.

The semantics of the dynamic modality $[\beta]\mathcal{C}$ is classical; it checks that the clause $\mathcal{C}$ holds at all β-reachable states. The reachability function ρ is extended to all compound actions β. Special for $\mathcal{CL}$ is that we extend the regular actions with synchrony and thus we define the reachability function for compound actions using the associated automata $GNFA(\beta)$, in the style of APDL [21].

4 Properties

Proposition 2 (validities). *The following statements hold:*

$$\models \neg O_{\mathcal{C}}(\mathbf{0}) \quad (1)$$
$$\models O_{\mathcal{C}}(\mathbf{1}) \quad (2)$$
$$\models P(\alpha) \rightarrow \neg F_{\mathcal{C}}(\alpha) \quad (3)$$
$$\models O_{\mathcal{C}}(\alpha) \rightarrow P(\alpha) \quad (4)$$
if $\alpha = \alpha'$ *then* $\models O_{\mathcal{C}}(\alpha) \leftrightarrow O_{\mathcal{C}}(\alpha') \quad (5)$
$$\models O_{\mathcal{C}}(\alpha) \rightarrow \neg F_{\mathcal{C}}(\alpha) \quad (6)$$
$$\models O_{\mathcal{C}}(\alpha) \wedge O_{\mathcal{C}}(\alpha') \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \quad (7)$$
$$\models O_{\mathcal{C}}(\alpha \cdot \alpha') \leftrightarrow O_{\mathcal{C}}(\alpha) \wedge [[\alpha]] O_{\mathcal{C}}(\alpha') \quad (8)$$
$$\models F_{\mathcal{C}}(\alpha) \rightarrow F_{\mathcal{C}}(\alpha \times \alpha') \quad (9)$$
$$\models F_{\mathcal{C}}(\alpha + \alpha') \leftrightarrow F_{\mathcal{C}}(\alpha) \wedge F_{\mathcal{C}}(\alpha') \quad (10)$$
$$\models F_{\mathcal{C}}(\alpha \cdot \alpha') \leftrightarrow F_{\mathcal{C}}(\alpha) \vee \langle\langle \alpha \rangle\rangle F_{\mathcal{C}}(\alpha') \quad (11)$$
$$\models P(\alpha + \alpha') \leftrightarrow P(\alpha) \wedge P(\alpha') \quad (12)$$
$$\models P(\alpha \cdot \alpha') \leftrightarrow P(\alpha) \wedge [\alpha] P(\alpha') \quad (13)$$

The following point out conflicts that are avoided in $\mathcal{CL}$:

$$\models \neg(O_{\mathcal{C}}(\alpha) \wedge F_{\mathcal{C}}(\alpha)) \tag{14}$$

$$\models \neg(P(\alpha) \wedge F_{\mathcal{C}}(\alpha)) \tag{15}$$

$$\textit{if } \alpha \#_{\mathcal{C}} \alpha' \textit{ then } \models \neg(O_{\mathcal{C}}(\alpha) \wedge O_{\mathcal{C}}(\alpha')) \tag{16}$$

The symbols $[[\cdot]]$ and $\langle\langle\cdot\rangle\rangle$ are the corresponding of the dynamic modalities $[\cdot]$ and $\langle\cdot\rangle$ only that they consider any action bigger than the action inside the modality. With the semantics of Table 2 we cannot prove validity of expressions like $F(a) \wedge (\phi \rightarrow P(a))$ which may be intuitive for the reader; e.g. some action a is forbidden and only in exceptional cases (when φ holds) it is permitted. The formula is not satisfied in a model which has a state s s.t. $\bullet_a \in \varrho(s)$ (from $F(a)$) and where ϕ holds and thus from the semantics of $P(a)$ is required that $\bullet_a \notin \varrho(s)$ which is impossible. Nevertheless, the example can be modelled in $\mathcal{CL}$ as $(\neg\varphi \rightarrow F(a)) \wedge (\varphi \rightarrow P(a))$ which is also more natural.

The following result shows that the semantics avoids *unwanted implications*.

Proposition 3 (unwanted implications). *The following statements hold:*

$$\not\models O_{\mathcal{C}}(\alpha) \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \tag{17}$$

$$\not\models O_{\mathcal{C}}(\alpha \times \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \tag{18}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha \times \alpha') \tag{19}$$

$$\not\models O_{\mathcal{C}}(\alpha \times \alpha') \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{20}$$

$$\not\models O_{\mathcal{C}}(\alpha) \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{21}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \tag{22}$$

$$\not\models F_{\mathcal{C}}(\alpha \times \alpha') \rightarrow F_{\mathcal{C}}(\alpha) \tag{23}$$

$$\not\models P(\alpha \times \alpha') \rightarrow P(\alpha) \tag{24}$$

$$\not\models O_{\mathcal{C}}(\alpha) \oplus O_{\mathcal{C}}(\alpha') \rightarrow O_{\mathcal{C}}(\alpha + \alpha') \tag{25}$$

$$\not\models O_{\mathcal{C}}(\alpha + \alpha') \rightarrow O_{\mathcal{C}}(\alpha) \oplus O_{\mathcal{C}}(\alpha') \tag{26}$$

We show that $\mathcal{CL}$ is decidable by showing that it has the *finite model property*. We do this by first showing that $\mathcal{CL}$ has the *tree model property*. For proving the finite model property we use the *selection* technique [23, sec.2.3] because it is hard to use the filtration technique in our case as we do not know what are subformulas of an obligation of a complex action like $O_{\mathcal{C}}(a \cdot (b + c))$. (For proofs see technical report [22].)

Theorem 1 (decidability). *$\mathcal{CL}$ with the semantics of Table 2 is decidable.*

In Appendix A.1 we give the relation between the full semantics presented so far and a trace semantics for $\mathcal{CL}$ from [13]. The results hold in a slightly restricted setting which is due to the restrictions on $\mathcal{CL}$ coming from the trace semantics and the run-time monitoring setting where it is used.

5 Conclusion

Some technical details of the interpretation of the actions and of the $\mathcal{CL}$ formulas have been omitted due to space restrictions in favor of intuitive explanations; the interested reader can find these details in [22]. The $\mathcal{CL}$ logic presented here has a decidable satisfiability problem. The technical difficulties in the underlying semantics come from the many properties that the logic needs to capture. Some of the novelties of $\mathcal{CL}$ are the use of synchronous actions and the definitions of the conflict relation and the normative structures.

References

1. Owe, O., Schneider, G., Steffen, M.: Components, objects, and contracts. In: SAVCBS 2007, pp. 91–94. ACM Digital Library, Dubrovnik (2007)
2. van der Torre, L.: Contextual deontic logic: Normative agents, violations and independence. Ann. Math. Artif. Intell. 37(1-2), 33–63 (2003)
3. von Wright, G.H.: Deontic logic. Mind 60, 1–15 (1951)
4. Fischer, M.J., Ladner, R.E.: Propositional modal logic of programs. In: STOC 1977, pp. 286–294. ACM Press, New York (1977)
5. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
6. von Wright, G.H.: An Essay in Deontic Logic and the General Theory of Action. North Holland Publishing Co., Amsterdam (1968)
7. Segerberg, K.: A deontic logic of action. Studia Logica 41(2), 269–282 (1982)
8. Meyer, J.J.C.: A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. Notre Dame Journal of Formal Logic 29(1), 109–136 (1988)
9. Broersen, J., Wieringa, R., Meyer, J.J.C.: A fixed-point characterization of a deontic logic of regular action. Fundam. Inf. 48(2-3), 107–128 (2001)
10. Milner, R.: Calculi for synchrony and asynchrony. TCS 25, 267–310 (1983)
11. Berry, G.: The foundations of Esterel. In: Proof, language, and interaction: essays in honour of Robin Milner, pp. 425–454. MIT Press, Cambridge (2000)
12. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: Bonsangue, M.M., Johnsen, E.B. (eds.) FMOODS 2007. LNCS, vol. 4468, pp. 174–189. Springer, Heidelberg (2007)
13. Kyas, M., Prisacariu, C., Schneider, G.: Run-time monitoring of electronic contracts. In: Cha, S(S.), Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (eds.) ATVA 2008. LNCS, vol. 5311, pp. 397–407. Springer, Heidelberg (2008)
14. Van der Meyden, R.: Dynamic logic of permission, the. In: LICS 1990, pp. 72–78. IEEE Computer Society Press, Los Alamitos (1990)
15. Castro, P.F., Maibaum, T.: A complete and compact propositional deontic logic. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) ICTAC 2007. LNCS, vol. 4711, pp. 109–123. Springer, Heidelberg (2007)
16. Harel, D.: Recurring dominoes: Making the highly undecidable highly understandable. In: Karpinski, M. (ed.) FCT 1983. LNCS, vol. 158, pp. 177–194. Springer, Heidelberg (1983)
17. Peleg, D.: Concurrent dynamic logic. In: STOC 1985, pp. 232–239. ACM Press, New York (1985)
18. Prakken, H., Sergot, M.: Dyadic deontic logic and contrary-to-duty obligation. In: Defeasible Deontic Logic, pp. 223–262. Kluwer Academic Publishers, Dordrecht (1997)
19. Prisacariu, C.: Extending Kleene Algebra with Synchrony – technicalities. Technical Report 376, Univ. Oslo (2008)
20. Ben-Ari, M., Halpern, J.Y., Pnueli, A.: Finite models for deterministic propositional dynamic logic. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 249–263. Springer, Heidelberg (1981)
21. Harel, D., Sherman, R.: Propositional dynamic logic of flowcharts. In: Karpinski, M. (ed.) FCT 1983. LNCS, vol. 158, pp. 195–206. Springer, Heidelberg (1983)
22. Prisacariu, C., Schneider, G.: CL: A Logic for Reasoning about Legal Contracts –Semantics. Technical Report 371, Univ. Oslo (2008)
23. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press, Cambridge (2001)

A Additional Proofs and Definitions

This Appendix contains formal definitions for some of the concepts given in the paper, as well as more thorough presentation of the results. All these, and more detailed explanations, examples, and full proofs can be found in the technical report [22].

The Definition 9 of *action negation* and the tree interpretation of the deontic actions in Theorem 3 are based on a notion of *canonical form*.

Definition 6 (canonical form). *We say that an action α is in* canonical form *denoted by $\underline{\alpha}$ iff it has the following form:*

$$\underline{\alpha} = +_{i \in I} \ \alpha_{\times}^{i} \cdot \underline{\alpha^{i}}$$

where $\alpha_{\times}^{i} \in \mathcal{A}_B^{\times}$ and $\underline{\alpha^{i}} \in \mathcal{A}$ is an action in canonical form. The indexing set I is finite as $\alpha_{\times}^{i} \in \mathcal{A}_B^{\times}$ are finite; therefore there is a finite number of application of the $+$ combinator. Actions $\mathbf{0}$ and $\mathbf{1}$ are in canonical form.

Theorem 2 ([19]). *For every action α there exists a corresponding action $\underline{\alpha}$ in canonical form and equivalent to α.*

Definition 7 (rooted tree). *A* rooted tree *is an acyclic connected graph $(\mathcal{N}, \mathcal{E})$ with a designated node r called* root *node. $\mathcal{N}$ is the set of* nodes *and $\mathcal{E}$ is the set of* edges *(where an edge is an ordered pair of nodes (n, m)). We consider* rooted trees with labeled edges *and denote the labeled directed edges with (n, α, m) and the tree with $(\mathcal{N}, \mathcal{E}, \mathcal{A}_B)$. The labels $\alpha \in 2^{\mathcal{A}_B}$ are* sets of basic labels*; e.g. $\alpha_1 = \{a, b\}$ or $\alpha_2 = \{a\}$ with $a, b \in \mathcal{A}_B$. Labels are compared for set equality (or set inclusion). Note the special* empty set *label. We consider a special label Λ to stand for an impossible label. We restrict our presentation to* finite *rooted trees (i.e., there is no infinite path in the graph starting from the root node). The set of all such defined trees is denoted $\mathcal{T}$.*

Theorem 3 (interpretation of deontic actions). *For any action α there exists a tree representation corresponding to the canonical form $\underline{\alpha}$.*

Proof. The *representation* is an interpretation function $I : \mathcal{A} \to \mathcal{T}$ which interprets all action terms as trees. More precisely, given an arbitrary action of $\mathcal{A}$, the canonical form is computed first and then I generates the tree representation.

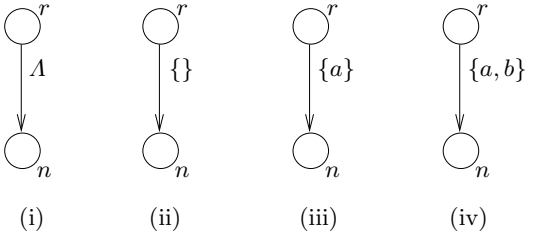


Fig. 1. Trees corresponding to $\mathbf{0}$, $\mathbf{1}$, $a \in \mathcal{A}_B$, and $a \times b \in \mathcal{A}_B^{\times}$

The function I is defined inductively. The basis is to interpret each concurrent action of $\mathcal{A}_B^{\times}$ as a tree with labeled edges from $2^{\mathcal{A}_B}$ as pictured in Fig.1. Note that actions of $\mathcal{A}_B^{\times} \cup \{\mathbf{0}, \mathbf{1}\}$ are in canonical form. For a general action in canonical form $\underline{\alpha} = +_{i \in I}\ \alpha_{\times}^i \cdot \underline{\alpha^i}$ the tree is generated by adding one branch to the root node for each element $\alpha_{\times}^i$ of the top summation operation. The label of the branch is the set $\{\alpha_{\times}^i\}$ corresponding to the concurrent action. The construction continues inductively by attaching at the end of each newly added branch the tree interpretation of the smaller action $\underline{\alpha^i}$.

The semantics for $O_{\mathcal{C}}$ relies on a notion of non-simulating reminder structure and on the operation of action negation for deontic actions.

Definition 8 (non-simulating reminder). *Whenever* $T\ \mathcal{S}_i\ K^{\mathcal{N}}$ *then we call the* maximal simulating structure w.r.t. T and i, *and denote it by* $K_{max}^{T,i} = (\mathcal{W}', \rho', \mathcal{V}', \varrho')$ *the sub-structure of* $K^{\mathcal{N}} = (\mathcal{W}, \rho, \mathcal{V}, \varrho)$ *s.t.:*

1. $i \in \mathcal{W}'$
2. $\mathcal{V}' = \mathcal{V}|_{\mathcal{W}'}$ *and* $\varrho' = \varrho|_{\mathcal{W}'}$
3. $\forall t \xrightarrow{\gamma} t' \in T$ *then* $\forall s \xrightarrow{\gamma'} s' \in K^{\mathcal{N}}$ *s.t.* $t\,\mathcal{S}\,s \wedge \gamma \subseteq \gamma' \wedge t'\,\mathcal{S}\,s'$ *do add* s' *to* $\mathcal{W}'$ *and add* $s \xrightarrow{\gamma'} s'$ *to* ρ'.

We call the non-simulating remainder of $K^{\mathcal{N}}$ w.r.t. T and i *the sub-structure* $K_{rem}^{T,i} = (\mathcal{W}'', \rho'', \mathcal{V}'', \varrho'')$ *of* $K^{\mathcal{N}}$ *s.t.:* $s \xrightarrow{\gamma} s' \in \rho''$ *iff* $s \xrightarrow{\gamma} s' \notin K_{max}^{T,i} \wedge s \in K_{max}^{T,i} \wedge \exists s \xrightarrow{\gamma} s'' \in K_{max}^{T,i}$; *and* $s \in \mathcal{W}''$ *iff* s *is part of a transition in* $K_{rem}^{T,i}$; *and* $\mathcal{V}'' = \mathcal{V}|_{\mathcal{W}''}$ *and* $\varrho'' = \varrho|_{\mathcal{W}''}$.

Definition 9 (action negation). *The* action negation *is denoted by* $\overline{\alpha}$ *and is defined as a function* $\overline{\ } : \mathcal{A} \to \mathcal{A}$ *(i.e. action negation is not a principal combinator for the actions) and works on the equivalent canonical form* $\underline{\alpha}$ *as:*

$$\overline{\alpha} = \overline{\underset{i \in I}{+}\ \alpha_{\times}^i \cdot \underline{\alpha^i}} = \underset{\beta_{\times} \in \overline{R}}{+} \beta_{\times}\ +\ \underset{j \in J}{+}\ \gamma_{\times}^j \cdot \overline{\underset{i \in I'}{+}\ \underline{\alpha^i}}$$

Consider $R = \{\alpha_{\times}^i \mid i \in I\}$. *The set* $\overline{R}$ *contains all the concurrent actions* $\beta_{\times}$ *with the property that* $\beta_{\times}$ *is not bigger than any of the actions* $\alpha_{\times}^i$:

$$\overline{R} = \{\beta_{\times} \mid \beta_{\times} \in \mathcal{A}_B^{\times} \text{ and } \forall i \in I, \alpha_{\times}^i \not\subset \beta_{\times}\};$$

and $\gamma_{\times}^j \in \mathcal{A}_B^{\times}$ *and* $\exists \alpha_{\times}^i \in R$ *s.t.* $\alpha_{\times}^i \subseteq \gamma_{\times}^j$. *The indexing set* $I' \subseteq I$ *is defined for each* $j \in J$ *as:*

$$I' = \{i \in I \mid \alpha_{\times}^i \subseteq \gamma_{\times}^j\}.$$

A.1 Relations with the Trace Semantics of $\mathcal{CL}$

In [13] we presented a trace semantics for $\mathcal{CL}$ with the goal of monitoring electronic contracts at run-time. This semantics is intended for identifying the *respecting* and *violating* traces of actions for a $\mathcal{CL}$ clause. Here we just present

Table 3. Trace semantics of $\mathcal{CL}$

For $\top, \bot, \rightarrow, \wedge, \vee, \oplus$	take a standard LTL-style semantics.
$\sigma \models [\alpha_\times]\mathcal{C}$	if $\alpha_\times = \sigma(0)$ and $\sigma(1..) \models \mathcal{C}$, or $\alpha_\times \neq \sigma(0)$.
$\sigma \models [\beta \cdot \beta']\mathcal{C}$	if $\sigma \models [\beta][\beta']\mathcal{C}$.
$\sigma \models [\beta + \beta']\mathcal{C}$	if $\sigma \models [\beta]\mathcal{C}$ and $\sigma \models [\beta']\mathcal{C}$.
$\sigma \models [\beta^*]\mathcal{C}$	if $\sigma \models \mathcal{C}$ and $\sigma \models [\beta][\beta^*]\mathcal{C}$.
$\sigma \models [\mathcal{C}_1?]\mathcal{C}_2$	if $\sigma \not\models \mathcal{C}_1$, or $if \sigma \models \mathcal{C}_1$ and $\sigma \models \mathcal{C}_2$.
$\sigma \models O_\mathcal{C}(\alpha_\times)$	if $\alpha_\times \subseteq \sigma(0)$, or if $\sigma(1..) \models \mathcal{C}$.
$\sigma \models O_\mathcal{C}(\alpha \cdot \alpha')$	if $\sigma \models O_\mathcal{C}(\alpha)$ and $\sigma \models [[\alpha]]O_\mathcal{C}(\alpha')$.
$\sigma \models O_\mathcal{C}(\alpha + \alpha')$	if $\sigma \models O_\bot(\alpha)$ or $\sigma \models O_\bot(\alpha')$ or $\sigma \models [\overline{\alpha + \alpha'}]\mathcal{C}$.
$\sigma \models F_\mathcal{C}(\alpha_\times)$	if $\alpha_\times \not\subseteq \sigma(0)$, or if $\alpha_\times \subseteq \sigma(0)$ and $\sigma(1..) \models \mathcal{C}$.
$\sigma \models F_\mathcal{C}(\alpha \cdot \alpha')$	if $\sigma \models F_\bot(\alpha)$ or $\sigma \models [[\alpha]]F_\mathcal{C}(\alpha')$.
$\sigma \models F_\mathcal{C}(\alpha + \alpha')$	if $\sigma \models F_\mathcal{C}(\alpha)$ and $\sigma \models F_\mathcal{C}(\alpha')$.

briefly the trace semantics and then concentrate on relating it with the full semantics of Table 2.

Consider an infinite *trace* denoted $\sigma = a_0, a_1, \ldots$ as a map $\sigma : \mathbb{N} \rightarrow \mathcal{A}_B^\times \cup \{\mathbf{1}\}$ from natural numbers (denoting positions) to concurrent actions from $\mathcal{A}_B^\times$. We denote by $\sigma(i)$ the *element* of a trace at position i, by $\sigma(i..j)$ a finite *subtrace*, and by $\sigma(i..)$ the infinite subtrace starting at position i in σ.

Consider the *satisfaction relation* $\models_t$ defined over pairs $(\sigma, \mathcal{C})$ of a trace and a contract which we write $\sigma \models_t \mathcal{C}$ and read as "trace σ *respects* the contract (clause) $\mathcal{C}$". For a brief definition of $\models_t$ see Table 3 and for details see [13].

The standard interpretation of $[\alpha_\times]\mathcal{C}$ is over branching structures as we did in Table 2. Here we interpret the two dynamic modalities over linear structures, i.e. over traces. A trace σ respects the formula $[\alpha_\times]\mathcal{C}$ if either the first (set of actions) element of the trace $\sigma(0)$ is not equal with the (set of basic actions forming the) action $\alpha_\times$ or otherwise $\sigma(0)$ is the same as action $\alpha_\times$ and $\mathcal{C}$ is respected by the rest of the trace (i.e. $\sigma(1..) \models \mathcal{C}$).

A trace σ respects an obligation $O_\mathcal{C}(\alpha_\times)$ if any of the two complementary conditions is satisfied. The first condition deals with the obligation itself: $O(\alpha_\times)$ is respected if the first action of the trace includes $\alpha_\times$. Otherwise, in case the obligation is violated,[2] the only way to fulfill the contract is by respecting the reparation $\mathcal{C}$; i.e. $\sigma(1..) \models \mathcal{C}$.

A folk technique called *linearization* takes (in our case) a pointed normative structure and returns all the (in)finite traces that start in the designated state i of the pointed structure. Denote this set of traces by $\|K^\mathcal{N}, i\|$. We use the notation $\sigma \in \|K^\mathcal{N}, i\|$ to mean that the trace σ is part of the traces of $K^\mathcal{N}$ starting as state i. Therefore, the following statement is obvious: *For any trace σ we can find a normative structure $K^\mathcal{N}$ and a state i s.t. $\sigma \in \|K^\mathcal{N}, i\|$.*

[2] Violation of an obligatory action is encoded by the action negation.

When not mentioned otherwise, the following results hold for a restricted syntax of $\mathcal{CL}$ which does not consider negation of clauses, nor tests inside the dynamic modalities. These syntactic restrictions are enough for doing run-time monitoring. For proofs see technical report [22].

Lemma 1. *For any* $K^{\mathcal{N}}$ *and* $i \in K^{\mathcal{N}}$, *if* $K^{\mathcal{N}}, i \models \mathcal{C}$ *then* $\forall \sigma \in \|K^{\mathcal{N}}, i\|\ \sigma \models_t \mathcal{C}$.

Corollary 1.

$$\bigcup_{K^{\mathcal{N}}, i \models \mathcal{C}} \|K^{\mathcal{N}}, i\| \subseteq \{\sigma \mid \sigma \models_t \mathcal{C}\}$$

Proposition 4. *With the general syntax of* $\mathcal{CL}$ *from Table 1 we can find a contract clause* $\mathcal{C}$ *s.t.* $\exists \sigma$ *s.t.* $\sigma \models_t \mathcal{C}$ *and* $\nexists K^{\mathcal{N}}, \nexists i \in K^{\mathcal{N}}$ *s.t.* $K^{\mathcal{N}}, i \models \mathcal{C}$.

Lemma 2 states the relation between the satisfiability in the trace semantics and satisfiability in the branching semantics. It says that if a contract clause is respected by some trace of actions then the contract is satisfiable in the branching semantics.

Lemma 2. *If* $\exists \sigma$ *s.t.* $\sigma \models_t \mathcal{C}$ *then* $\exists K^{\mathcal{N}}, \exists i \in K^{\mathcal{N}}$ *s.t.* $K^{\mathcal{N}}, i \models \mathcal{C}$.

Lemma 3. *If* $\sigma \models_t \mathcal{C}$ *then* $\exists K^{\mathcal{N}}, \exists i \in K^{\mathcal{N}}$ *s.t.* $K^{\mathcal{N}}, i \models \mathcal{C}$ *and* $\sigma \in \|K^{\mathcal{N}}, i\|$.

Corollary 2.

$$\{\sigma \mid \sigma \models_t \mathcal{C}\} \subseteq \bigcup_{K^{\mathcal{N}}, i \models \mathcal{C}} \|K^{\mathcal{N}}, i\|$$

The two corollaries relate the validities in the two semantics (under the restricted syntax). If a clause $\mathcal{C}$ is valid w.r.t. the trace semantics (i.e. there is no way of doing a sequence of actions to violate the contract) then the clause is valid in the full semantics (i.e. any model of the contract also entails the existence of this contract clause).

Ehrenfeucht-Fraïssé Games on Random Structures

Benjamin Rossman*

Massachusetts Institute of Technology, Cambridge MA 02139, USA

Abstract. Certain results in circuit complexity (e.g., the theorem that AC^0 functions have low average sensitivity [5, 17]) imply the existence of winning strategies in Ehrenfeucht-Fraïssé games on pairs of random structures (e.g., ordered random graphs $G = G(n, 1/2)$ and $G^+ = G \cup \{\text{random edge}\}$). Standard probabilistic methods in circuit complexity (e.g., the Switching Lemma [11] or Razborov-Smolensky Method [19, 21]), however, give no information about how a winning strategy might look. In this paper, we attempt to identify specific winning strategies in these games (as explicitly as possible). For random structures G and G^+, we prove that the *composition of minimal strategies* in r-round Ehrenfeucht-Fraïssé games $\eth_r(G, G)$ and $\eth_r(G^+, G^+)$ is almost surely a winning strategy in the game $\eth_r(G, G^+)$. We also examine a result of [20] that ordered random graphs $H = G(n, p)$ and $H^+ = H \cup \{\text{random } k\text{-clique}\}$ with $p(n) \ll n^{-2/(k-1)}$ (below the k-clique threshold) are almost surely indistinguishable by $\lfloor k/4 \rfloor$-variable first-order sentences of any fixed quantifier-rank r. We describe a winning strategy in the corresponding r-round $\lfloor k/4 \rfloor$-pebble game using a technique that combines strategies from several auxiliary games.

1 Introduction

Let $\mathcal{A}$ be an arbitrary finite structure, let P be a uniform random subset of A, let q be a uniform random element of A, and let $P' = P \vartriangle \{q\}$. Let $(\mathcal{A}, P)$ and $(\mathcal{A}, P')$ denote the expansions of $\mathcal{A}$ by a new unary relation symbol interpreted as P and P', respectively. Results in circuit complexity [1, 8] and descriptive complexity [12, 10] imply that structures $(\mathcal{A}, P)$ and $(\mathcal{A}, P')$ almost surely satisfy the same first-order sentences up to a fixed quantifier-rank r (as the size of $\mathcal{A}$ increases). Equivalently, there exists a winning strategy (for "Duplicator") in the r-round Ehrenfeucht-Fraïssé game on these structures, which we denote by $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$.

Standard proofs of this fact via circuit complexity merely show that a winning strategy must exist using probabilistic arguments (generally based on either the Switching Lemma [4, 11] or Razborov-Smolensky Method [19, 21]). These proofs, however, say nothing about what a winning strategy might actually look like. In this paper, we aim for an explicit characterization of a winning strategy in $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$. Our result involves two natural notions concerning strategies:

* Supported by a National Defense Science and Engineering Graduate Fellowship.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 350–364, 2009.

composition (Definition 6) given strategies $\mathbf{S}_1$ and $\mathbf{S}_2$ in games $\eth_r(\mathcal{A}, \mathcal{B}_1)$ and $\eth_r(\mathcal{B}_2, \mathcal{C})$ (where structures $\mathcal{B}_1$ and $\mathcal{B}_2$ have a common universe), there is a natural composition strategy $\mathbf{S}_1 \circ \mathbf{S}_2$ in the game $\eth_r(\mathcal{A}, \mathcal{C})$

minimal strategy (Definition 7) for every structure $\mathcal{A}$ with universe $\{1, \ldots, n\}$, there is a canonical (lexicographically) minimal winning strategy in the game $\eth_r(\mathcal{A}, \mathcal{A})$.

We show (Theorem 8) that the composition $\mathbf{S} \circ \mathbf{S}'$ is almost surely a winning strategy in the game $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$ where $\mathbf{S}$ and $\mathbf{S}'$ are the minimal winning strategies in games $\eth_r((\mathcal{A}, P), (\mathcal{A}, P))$ and $\eth_r((\mathcal{A}, P'), (\mathcal{A}, P'))$.

A second result of this paper (Proposition 13) gives a criterion for (constructively) establishing the existence of winning strategies in r-round k-pebble games among a family of k-tupled structures with a common universe. We explain how this criterion is applied in [20] to prove that k-CLIQUE is not definable in $\lfloor k/4 \rfloor$-variable first-order logic. This criterion involves a novel technique for combining games across multiple structures.

Organization of the paper. In §2, we give the relevant definitions of structures, games and strategies (including minimal strategies and composition of strategies). In §3, we identify a particular winning strategy in the game $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$. In §4, we present a criterion for $\equiv^k_r$-equivalence. In §5, we explain how this criterion is used to prove that k-CLIQUE is not definable in $\lfloor k/4 \rfloor$-variable first-order logic. In §6, we study this criterion in the setting of games on Kripke structures. Two appendices (§A, §B) contains proofs that do not fit in the body of the paper.

2 Preliminaries: Structures, Games and Strategies

Throughout this paper, *structures* are finite relational structures. We assume that the universe of every structure is equipped with a linear order (not necessarily occurring as a relation in the structure); without loss of generally, every structure has universe $\{1, \ldots, n\}$ for some natural number n.

Definition 1 (STRUCTURES). *We now make this precise. A* relational signature σ *consists of finitely many "relation symbols"* $R_1, \ldots, R_t$ *with associated "arities"* $r_1, \ldots, r_t \geqslant 1$. *A* σ-structure $\mathcal{A}$ *consists of a set* A *(called the* universe of $\mathcal{A}$*) together with interpretations for all relation symbols in* σ*, i.e., relations* $R_i^{\mathcal{A}} \subseteq A^{r_i}$ *for* $i = 1, \ldots, t$. *We denote the universe of* $\mathcal{A}$ *by* A*, the universe of* $\mathcal{B}$ *by* B*, etc. For* $S \subseteq A$*, the induced substructure of* $\mathcal{A}$ *with universe* S *is denoted by* $\mathcal{A}|_S$.

By default, all structures are σ*-structure for an arbitrary fixed relational signature* σ. *If* $\mathcal{A}$ *is a structure and* $R \subseteq A^r$ *is an* r*-ary relation on* A*, then we denote by* $(\mathcal{A}, R)$ *the* $(\sigma \cup \{\underline{R}\})$*-structure expanding* $\mathcal{A}$*, in which* R *interprets a new* r*-ary relational symbol* $\underline{R}$.

A k-tupled structure *is a pair* $(\mathcal{A}, \bar{a})$ *where* $\bar{a} = (a_1, \ldots, a_k) \in A^k$. *In particular, structures are* 0*-tupled structures.*

Definition 2 ($\equiv_r$- AND $\equiv_r^k$-EQUIVALENCE). *For all $k, r \in \mathbb{N}$, equivalence relations $\equiv_r$ and $\equiv_r^k$ on the class of k-tupled structures are defined by the following induction. Let $(\mathcal{A}, \bar{a})$ and $(\mathcal{B}, \bar{b})$ be k-tupled structures.*

- $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b}) \iff (\mathcal{A}, \bar{a}) \equiv_0^k (\mathcal{B}, \bar{b}) \overset{def}{\iff} \{(a_1, b_1), \ldots, (a_k, b_k)\}$ *is a partial isomorphism between $\mathcal{A}$ and $\mathcal{B}$ (i.e., a bijective function between subsets of A and B that preserves all relations).*

For $r \geqslant 1$,

- $(\mathcal{A}, \bar{a}) \equiv_r (\mathcal{B}, \bar{b}) \overset{def}{\iff} \begin{cases} \forall a' \in A \ \exists b' \in B \ \ (\mathcal{A}, \bar{a}a') \equiv_{r-1} (\mathcal{B}, \bar{b}b'), \\ \forall b' \in B \ \exists a' \in A \ \ (\mathcal{A}, \bar{a}a') \equiv_{r-1} (\mathcal{B}, \bar{b}b'); \end{cases}$

  (note: here $\equiv_{r-1}$ is an equivalence relation on $(k+1)$-tupled structures)
- $(\mathcal{A}, \bar{a}) \equiv_r^k (\mathcal{B}, \bar{b}) \overset{def}{\iff} (\mathcal{A}, \bar{a}) \equiv_0^k (\mathcal{B}, \bar{b})$ *and for all $i \in [k]$,*

$$\forall a' \in A \ \exists b' \in B \ (\mathcal{A}, a_1, \ldots, a_{i-1}, a', a_{i+1}, \ldots, a_k) \equiv_{r-1}^k (\mathcal{B}, b_1, \ldots, b_{i-1}, b', b_{i+1}, \ldots, b_k),$$
$$\forall b' \in B \ \exists a' \in A \ (\mathcal{A}, a_1, \ldots, a_{i-1}, a', a_{i+1}, \ldots, a_k) \equiv_{r-1}^k (\mathcal{B}, b_1, \ldots, b_{i-1}, b', b_{i+1}, \ldots, b_k).$$

This induction is clearly well-founded: the definition of $\equiv_r$ on k-tupled structures depends on the definition of $\equiv_{r-1}$ on $(k+1)$-tupled structures, etc., which eventually depends on the (base case) definition of $\equiv_0$ on $(k+r)$-tupled structures. Note that $\equiv_{r+1}$ refines $\equiv_r$, which refines $\equiv_r^k$.

Definition 3 (EHRENFEUCHT-FRAÏSSÉ GAME). *The* r-round Ehrenfeucht-Fraïssé game on structures $\mathcal{A}$ and $\mathcal{B}$, *denoted $\Im_r(\mathcal{A}, \mathcal{B})$, is played as follows. There are two players, Spoiler (who wishes to establish that $\mathcal{A}$ and $\mathcal{B}$ are non-isomorphic) and Duplicator (who attempts to prevent Spoiler from achieving his goal). The "game board" consists of structures $\mathcal{A}$ and $\mathcal{B}$ and the "game pieces" are r pairs of pebbles $(p_1, q_1), \ldots, (p_r, q_r)$. The game is played in a sequence of r rounds. In round i of the game, Spoiler picks a structure (either $\mathcal{A}$ or $\mathcal{B}$) and places pebble p_i on an element of his choice in that structure. Duplicator then replies by placing pebble q_i on an element of his choice in the other structure. After r rounds, the positions of pebbles $p_1, \ldots, p_k, q_1, \ldots, q_k$ describe two r-tuples $(a_1, \ldots, a_r) \in A^r$ and $(b_1, \ldots, b_r) \in B^r$. Duplicator is declared the winner if and only if $\{(a_1, b_1), \ldots, (a_r, b_r)\}$ is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ (i.e., an isomorphism from $\mathcal{A}|_{\{a_1,\ldots,a_r\}}$ to $\mathcal{B}|_{\{b_1,\ldots,b_r\}}$).*

The r-round k-pebble game on k-tupled structures $(\mathcal{A}, \bar{a})$ and $(\mathcal{B}, \bar{b})$, *denoted $\Im_r^k((\mathcal{A}, \bar{a}), (\mathcal{B}, \bar{b}))$, is similar. However, in this game there are exactly k pairs of pebbles $(\alpha_1, \beta_1), \ldots, (\alpha_k, \beta_k)$. At the start of the game, these pairs of pebbles rest on $(a_1, b_1), \ldots, (a_k, b_k)$. In each round of the game, Spoiler picks an index $j \in [k]$ and a structure (either $\mathcal{A}$ or $\mathcal{B}$) and moves the jth pebble in that structure to an element of his choice. Duplicator then moves the jth pebble in the other structure to an element of his choice. Duplicator is declared the winner if and only if for every $i \in [r]$, the set $\{(\alpha_1^{(i)}, \beta_1^{(i)}), \ldots, (\alpha_k^{(i)}, \beta_k^{(i)})\}$ is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ where $\alpha_j^{(i)}, \beta_j^{(i)}$ denote the positions in A, B of pebbles α_j, β_j after round i of the game.*

We are interested in strategies in games $\Im_r(\mathcal{A},\mathcal{B})$ and $\Im_r^k((\mathcal{A},\bar{a}),(\mathcal{B},\bar{b}))$. By "strategy" we always mean a *deterministic strategy for Duplicator*; we are never concerned with strategies for Spoiler, who we simply assume plays optimally. For instance, the statement "there exists a winning strategy" should be read as "there exists a (deterministic) winning strategy (for Duplicator)".

To avoid redundancy, we present definitions and lemmas concerning the game $\Im_r(\mathcal{A},\mathcal{B})$ only. It will be obvious how to adapt the corresponding definitions and lemmas to the k-pebble game $\Im_r^k((\mathcal{A},\bar{a}),(\mathcal{B},\bar{b}))$.

Definition 4 (Strategy). *An r-round strategy on sets A and B is a function*

$$\mathbf{S} : \bigcup_{i=1}^{r}(A \sqcup B)^i \longrightarrow A \sqcup B$$

such that $\mathbf{S}(x_1,\ldots,x_i) \in A \iff x_i \in B$ for all $(x_1,\ldots,x_i) \in (A \sqcup B)^i$. The intention is that if $x_1,\ldots,x_i$ are Spoiler's moves in the first i rounds (i.e., which elements of which structures he plays), then $\mathbf{S}(x_1,\ldots,x_i)$ is Duplicator's response in round i under strategy $\mathbf{S}$.

We say that $\mathbf{S}$ is a winning strategy *in the game $\Im_r(\mathcal{A},\mathcal{B})$ if Duplicator is guaranteed to win by playing according to $\mathbf{S}$ (no matter how Spoiler plays).*

Note that we define strategies on pairs of sets A and B, rather than structures $\mathcal{A}$ and $\mathcal{B}$. For structures $\mathcal{A}_1,\mathcal{A}_2,\mathcal{B}_1,\mathcal{B}_2$ where $A_1 = A_2$ and $B_1 = B_2$, the same strategy $\mathbf{S}$ might thus be a winning strategy in $\Im_r(\mathcal{A}_1,\mathcal{B}_1)$, but not in $\Im_r(\mathcal{A}_2,\mathcal{B}_2)$.

The simplest example of a winning strategy is the "copycat" strategy in the r-round game on two copies of a single structure $\mathcal{A}$. Duplicator easily defeats Spoiler by always playing the same element in the opposite structure.

The next proposition (a basic fact in model theory, see e.g. [16]) connects games, $\equiv_r^{(k)}$-equivalence and logic.

Proposition 5. *The following are equivalent:*

i. $\mathcal{A} \equiv_r \mathcal{B}$ *(resp.* $(\mathcal{A},\bar{a}) \equiv_r^k (\mathcal{B},\bar{b})$*);*
ii. *there exists a winning strategy in* $\Im_r(\mathcal{A},\mathcal{B})$ *(resp.* $\Im_r^k((\mathcal{A},\bar{a}),(\mathcal{B},\bar{b}))$*);*
iii. *$\mathcal{A}$ and $\mathcal{B}$ satisfy the same first-order sentences of quantifier-rank r (resp. $(\mathcal{A},\bar{a})$ and $(\mathcal{B},\bar{b})$ satisfy the same first-order formulas $\phi(x_1,\ldots,x_k)$ of quantifier-rank r in which every subformula has at most k free variables).*

The following concept of the composition of strategies is fairly intuitive.

Definition 6 (Composition of strategies). *Let $\mathbf{S}$ be an r-round strategy on sets A and B, and let $\mathbf{T}$ be an r-round strategy on sets B and C. The* composition *$\mathbf{S} \circ \mathbf{T}$ is an r-round strategy on A and C defined as follows. Given $i \in \{1,\ldots,r\}$ and $(x_1,\ldots,x_i) \in (A \sqcup C)^i$, define $(y_1,\ldots,y_i) \in (A \sqcup B)^i$ and $(z_1,\ldots,z_i) \in (B \sqcup C)^i$ inductively for $j = 1,\ldots,i$ by*

- *if $x_j \in A$, then $y_j \triangleq x_j$ and $z_j \triangleq \mathbf{T}(z_1,\ldots,z_{j-1},\mathbf{S}(y_1,\ldots,y_j))$,*
- *if $x_j \in C$, then $z_j \triangleq x_j$ and $y_j \triangleq \mathbf{S}(y_1,\ldots,y_{j-1},\mathbf{T}(z_1,\ldots,z_j))$.*

Then $(\mathbf{S} \circ \mathbf{T})(x_1,\ldots,x_i) \triangleq \begin{cases} z_i & \text{if } x_i \in A, \\ y_i & \text{if } x_i \in C. \end{cases}$

For two structures $\mathcal{A}$ and $\mathcal{B}$ with given well-orderings such that $\mathcal{A} \equiv_r \mathcal{B}$, there is a canonical way to define a winning strategy in the r-round game on $\mathcal{A}$ and $\mathcal{B}$: let Duplicator always play the minimal winning move.

Definition 7 (Minimal winning strategy). *Let $\mathcal{A}$ and $\mathcal{B}$ be structures with given well-orderings such that $\mathcal{A} \equiv_r \mathcal{B}$. The* (lexicographically) minimal winning r-round strategy $\mathbf{M}_r^{\mathcal{A},\mathcal{B}} : \bigcup_{i=1}^{r} (A \sqcup B)^i \longrightarrow (A \sqcup B)$ *on $\mathcal{A}$ and $\mathcal{B}$ is defined inductively as follows. Let $i \in \{1, \ldots, r\}$ and assume $\mathbf{M}_r^{\mathcal{A},\mathcal{B}}$ has been defined on all sequences of length $i-1$. Consider any $(x_1, \ldots, x_i) \in (A \sqcup B)^i$. For all $j \in \{1, \ldots, i-1\}$, define $a_j \in A$ and $b_j \in B$ by*

$$a_j = \begin{cases} x_j & \text{if } x_j \in A, \\ \mathbf{M}_r^{\mathcal{A},\mathcal{B}}(x_1, \ldots, x_j) & \text{if } x_j \in B, \end{cases} \qquad b_j = \begin{cases} x_j & \text{if } x_j \in B, \\ \mathbf{M}_r^{\mathcal{A},\mathcal{B}}(x_1, \ldots, x_j) & \text{if } x_j \in A. \end{cases}$$

Assuming that $(\mathcal{A}, a_1, \ldots, a_{i-1}) \equiv_{r-i+1} (\mathcal{B}, b_1, \ldots, b_{i-1})$ (which is guaranteed by the induction), define $\mathbf{M}_r^{\mathcal{A},\mathcal{B}}(x_1, \ldots, x_i) \in A \sqcup B$ as follows:

- *if $x_i \in A$, then $\mathbf{M}_r^{\mathcal{A},\mathcal{B}}(x_1, \ldots, x_i)$ is the minimal $b_i \in B$ such that*
$$(\mathcal{A}, a_1, \ldots, a_{i-1}, x_i) \equiv_{r-i} (\mathcal{B}, b_1, \ldots, b_{i-1}, b_i),$$
- *if $x_i \in B$, then $\mathbf{M}_r^{\mathcal{A},\mathcal{B}}(x_1, \ldots, x_i)$ is the minimal $a_i \in A$ such that*
$$(\mathcal{A}, a_1, \ldots, a_{i-1}, a_i) \equiv_{r-i} (\mathcal{B}, b_1, \ldots, b_{i-1}, x_i).$$

We write $\mathbf{M}_r^{\mathcal{A}}$ (instead of $\mathbf{M}_r^{\mathcal{A},\mathcal{A}}$) for the minimal winning strategy in the r-round game on two copies of a single structure $\mathcal{A}$. (Note that $\mathbf{M}_r^{\mathcal{A}}$ is generally not the "copycat" strategy.)

3 A Winning Strategy in $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$

We now return the result mentioned in §1 concerning a winning strategy in the game $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$. As in §1, let $\mathcal{A}$ be an arbitrary structure with a universe of size n, let P be a uniform random subset of A, let q be a uniform random element of A, and let $P' = P \vartriangle \{q\}$. Let $(\mathcal{A}, P)$ and $(\mathcal{A}, P')$ denote the expansions of $\mathcal{A}$ by a new unary relation symbol interpreted as P and P', respectively.

Theorem 8. *The composition of minimal strategies $\mathbf{M}_r^{(\mathcal{A},P)} \circ \mathbf{M}_r^{(\mathcal{A},P')}$ is almost surely a winning strategy in the game $\eth_r((\mathcal{A}, P), (\mathcal{A}, P'))$. In fact,*

$$\Pr\left[\mathbf{M}_r^{(\mathcal{A},P)} \circ \mathbf{M}_r^{(\mathcal{A},P')} \text{ is not winning in } \eth_r((\mathcal{A}, P), (\mathcal{A}, P'))\right] \leqslant O\big((\log n)^{O(r)}/n\big).$$

The key notion in proving Theorem 8 is that of an $\mathcal{A}$-*minimal* r-tuple. We will show (Lemma 12) that the set of induced substructures on $\mathcal{A}$-minimal r-tuples contains the essential information about the strategy $\mathbf{M}_r^{\mathcal{A}}$.

Definition 9 ($\mathcal{A}$-Minimal tuples). *An r-tuple $(a_1, \ldots, a_r) \in A^r$ is* $\mathcal{A}$-minimal *if for all $i \in \{1, \ldots, r\}$, there exists $a' \in A$ such that $a_i = \mathbf{M}_r^{\mathcal{A}}(a_1, \ldots, a_{i-1}, a')$.*

The next lemma is essentially a folklore result in model theory. (Lemma 10 is also valid when $\mathcal{A}$ is an infinite structure with a given well-ordering.)

Lemma 10. *There exists a constant $c = c(r, \sigma)$ (depending only on r and the signature σ) such that for every structure $\mathcal{A}$, there exist $\leqslant c$ distinct $\mathcal{A}$-minimal r-tuples.*

Proof (sketch). This lemma follows from the folklore fact that there are only finitely many $\equiv_r$-equivalence classes of k-tupled structures for all $r, k \in \mathbb{N}$ (see [16]). (In fact, one can prove a bound of $c \leqslant \prod_{j=1}^{r} c_j$ where c_j is the number of $\equiv_{r-j}$-equivalence classes of $(k+j)$-tupled structures.)

Definition 11 (Strong r-equivalence). *Structures $\mathcal{A}_1$ and $\mathcal{A}_2$ with a common universe A are* strongly r-equivalent *if for every $(a_1, \ldots, a_r) \in A^r$,*

- *$(a_1, \ldots, a_r)$ is $\mathcal{A}_1$-minimal if and only if it is $\mathcal{A}_2$-minimal, and*
- *if $(a_1, \ldots, a_r)$ is $\mathcal{A}_1$-minimal, then $\mathcal{A}_1|_{\{a_1,\ldots,a_r\}} = \mathcal{A}_2|_{\{a_1,\ldots,a_r\}}$ (i.e., these induced substructures are identical).*

Alternatively, can replace these two conditions with the single (equivalent) condition that $\mathcal{A}_1|_{\{a_1,\ldots,a_r\}} = \mathcal{A}_2|_{\{a_1,\ldots,a_r\}}$ whenever $(a_1, \ldots, a_r)$ is $\mathcal{A}_1$-minimal or $\mathcal{A}_2$-minimal.

The next lemma characterizes strong equivalence via minimal strategies.

Lemma 12. *$\mathcal{A}_1$ and $\mathcal{A}_2$ are strongly r-equivalent if and only if $\mathbf{M}_r^{\mathcal{A}_1} \circ \mathbf{M}_r^{\mathcal{A}_2}$ is a winning strategy in $\eth_r(\mathcal{A}_1, \mathcal{A}_2)$.*

We omit the proof of Lemma 12, which follows easily from definitions. Note that Lemma 12 reduces Theorem 8 to the inequality

$$\Pr\big[(\mathcal{A}, P) \text{ and } (\mathcal{A}, P') \text{ are not strongly } r\text{-equivalent}\big] \leqslant O\big((\log n)^{O(r)}/n\big).$$

The remainder of the proof of Theorem 8 (which proves this inequality using a result from circuit complexity that AC^0 functions have low average sensitivity) is given in Appendix A.

4 Criterion for $\equiv_r^k$-Equivalence

We present a criterion for establishing $\equiv_r^k$-equivalences among a family of k-tupled structures with the same universe. The criterion speaks about a family $\{\mathcal{G}_{\bar{v}}\}_{\bar{v} \in V^k}$ of structures indexed by k-tuples over a common universe V, plus an additional structure $\mathcal{G}^*$ with universe V. We give two hypotheses which together imply that $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_r^k (\mathcal{G}^*, \bar{v})$ for every k-tuple $\bar{v} \in V^k$. In the next section, we describe how this condition is used to show that ℓ-Clique is not definable in $\lfloor \ell/4 \rfloor$-variable first-order logic.

To state the criterion, we fix a set V and a symmetric binary relation $\sim$ on V with finite diameter d (so that every two elements of V are connected by a $\sim$-path of length $\leqslant d$). For k-tuples $\bar{v}, \bar{w} \in V^k$, let

$$\bar{v} \sim \bar{w} \iff^{\text{def}} \exists i \in [k] \text{ such that } v_i \sim w_i \text{ and } v_j = w_j \text{ for all } j \in [k] \setminus \{i\}.$$

(In other words, the binary relation $\sim$ on k-tuples is the kth Cartesian power of binary relation $\sim$ on V.)

Proposition 13 (Criterion for $\equiv_r^k$). *Let $\{\mathcal{G}_{\bar{v}}\}_{\bar{v} \in V^k}$ be a family of structures $\mathcal{G}_{\bar{v}}$ with universe V (indexed by k-tuples $\bar{v} \in V^k$) and let $\mathcal{G}^*$ be another structure with universe V. Suppose that*

i. $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_0 (\mathcal{G}^*, \bar{v})$ *for all* $\bar{v} \in V^k$, *and*
ii. $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_{rd} (\mathcal{G}_{\bar{w}}, \bar{v})$ *for all* $\bar{v}, \bar{w} \in V^k$ *such that* $\bar{v} \sim \bar{w}$.

Then $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_r^k (\mathcal{G}^*, \bar{v})$ for all $\bar{v} \in V^k$.

Remark 14. *Proposition 13 is valid with a weaker hypothesis in which $\equiv_{rd}$ is replaced by $\equiv_{rd}^k$. It is even valid when V is infinite. However, Proposition 13 as stated is all that we require (for the application in §5).*

Proposition 13 can be proved by a very simple argument using the inductive definition of $\equiv_{rd}$. However, we prefer to understand Proposition 13 in light of the winning strategy it entails in the game $\eth_r^k((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$. An examination of the proof reveals a *deterministic strategy* for winning the game $\eth_r^k((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$ given black-box winning strategies in games $\eth_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}_{\bar{v}}, \bar{w}))$ for all $\bar{v}, \bar{w} \in V^k$ such that $\bar{v} \sim \bar{w}$. Under this strategy for $\eth_r^k((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$, Duplicator plays a sequence of simulated games $\eth_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}_{\bar{v}}, \bar{w}))$, in fact making $\binom{r}{2}$ queries to these auxiliary black-box strategies.

We will explicate this strategy (and also prove Proposition 13) later in §6. First, let's see a neat application of Proposition 13.

5 ℓ-Clique Requires $\lfloor \ell/4 \rfloor$ Variables

Fix any $\ell \geqslant 2$ and let $k = \lfloor \ell/4 \rfloor$.

Let $\mathcal{G} = (V, E, <)$ be an ordered Erdos-Renyi random graph where $V = [n]$ and E is a random anti-reflexive symmetric binary relation on V which includes each pair of vertices independently with probability $p(n) = n^{-2/(\ell-1.5)}$. Note that $p(n)$ is below the ℓ-clique threshold, i.e., $\mathcal{G}$ is almost surely ℓ-clique-free. Let A be a uniform random ℓ-element subset of V, let $\mathcal{G}^* = (\mathcal{G} \cup \text{clique on } A)$, and let $\bar{1} = (1, \ldots, 1)$ denote the all 1's k-tuple in V^k.

The following theorem is proved in [20].

Theorem 15. *For every r, it holds a.a.s. (asymptotically almost surely as $n \to \infty$) that $(\mathcal{G}, \bar{1}) \equiv_r^k (\mathcal{G}^*, \bar{1})$.*

Since $\mathcal{G}$ is almost surely ℓ-clique-free, while $\mathcal{G}^*$ contains an ℓ-clique (with probability 1), it follows that ℓ-clique is not definable in k-variable first-order logic.

We now explain how Theorem 15 is proved using Proposition 13. There are three steps.

Step 1. For a sufficiently small constant $\varepsilon > 0$ (to be determined), we fix an arbitrary reflexive and symmetric binary relation $\sim$ on V with degree $\leqslant n^{\varepsilon}$ and diameter $\leqslant 2/\varepsilon$ (e.g., a spanning tree).

Step 2. For all k-tuples $\bar{v} \in V^k$, define $A_{\bar{v}} \subseteq A$ by

$$A_{\bar{v}} \triangleq \left\{ a \in A : \exists A' \subseteq A \text{ s.t. } \begin{array}{l} |A'| < 2k \text{ and} \\ (\mathcal{G} \cup \text{clique on } A') \not\equiv_{\lceil 2r/\varepsilon \rceil} (\mathcal{G} \cup \text{clique on } A' \setminus \{a\}) \end{array} \right\}$$

and let $\mathcal{G}_{\bar{v}} \triangleq (\mathcal{G} \cup \text{clique on } A_{\bar{v}})$.

Step 3. We show that for all $\bar{v} \in V^k$,

$$\Pr\left[|A_{\bar{v}}| > 0\right] = o(1),$$
$$\Pr\left[|\textstyle\bigcup_{\bar{w} \sim \bar{v}} A_{\bar{w}}| \geqslant 2k\right] = o(1/n^k).$$

Steps 1 and 2 are merely definitions. Step 3 (where all the work is done) is proved by a probabilistic argument with ingredients from circuit complexity (see [20] for the proof). However, the intuition behind Step 3 is not hard to understand. The edge probability $p(n) = n^{-2/(\ell-1.5)}$ lies *below* the ℓ-clique threshold $\Theta(n^{-2/(\ell-1)})$), but *above* the $(\ell-1)$-clique threshold $\Theta(n^{-2/(\ell-2)})$. In particular, for every $\ell' < \ell$, the random graph $\mathcal{G}$ almost surely has *many* ℓ'-cliques. In particular, $\mathcal{G}$ almost surely has $\omega(n^k)$ cliques of size $2k$ ($\approx \ell/2$).

For every k-tuple $\bar{v} \in V^k$, the set $A_{\bar{v}}$ depends only on the $\equiv_{\lceil 2r/\varepsilon \rceil}$-equivalence classes of ordered graphs $(\mathcal{G} \cup \text{clique on } A')$ for subsets $A' \subseteq A$ of size $|A'| < 2k$. But because $\mathcal{G}$ already contains huge numbers of cliques of size $\leqslant 2k$, the addition of a random ℓ'-clique where $\ell' \leqslant 2k$ is *unlikely* to change the $\equiv_{\lceil 2r/\varepsilon \rceil}$-equivalence class of $(\mathcal{G}, \bar{v})$: this boils down to the fact that AC^0 functions have low average sensitivity (cp. Appendix 8). In fact, $\Pr[|A_{\bar{v}}| \geqslant \ell']$ is roughly bounded by $1/\mathrm{E}[\#\text{ of } \ell'\text{-cliques in } \mathcal{G}]$. Thus, we get $\Pr[|A_{\bar{a}}| > 0] = o(1)$ and $\Pr[|A_{\bar{v}}| \geqslant 2k] < o(1/n^k)$. Using the fact that $\bar{v}$ has at most kn^{ε} $\sim$-neighbors (and picking a sufficiently small constant ε in Step 1), we are able to show $\Pr[|\bigcup_{\bar{w} \sim \bar{v}} A_{\bar{w}}| \geqslant 2k] \leqslant o(1/n^k)$, proving Step 3.

Steps 1–3, together with Proposition 13, furnish a proof of Theorem 15. By Step 3, it holds almost surely that $A_{\bar{1}} = \emptyset$ (hence $\mathcal{G}_{\bar{1}} = \mathcal{G}$) and $|\bigcup_{\bar{w} \sim \bar{v}} A_{\bar{w}}| < 2k$ for all $\bar{v} \in V^k$ (taking a union bound over all k-tuples in V^k). Given that these events hold, hypotheses (i) and (ii) of Proposition 13 follow directly from the definition of $A_{\bar{v}}$ and $\mathcal{G}_{\bar{v}}$ in Step 2. Therefore, by Proposition 13, we have $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_r^k (\mathcal{G}^*, \bar{v})$ for all $\bar{v} \in V^k$. In particular, we have $(\mathcal{G}, \bar{1}) = (\mathcal{G}_{\bar{1}}, \bar{1}) \equiv_r^k (\mathcal{G}^*, \bar{1})$, proving Theorem 15.

6 Winning Strategy Behind the $\equiv_r^k$-Criterion (Prop. 13)

We now analyze the winning strategy in the game $\eth_r^k((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$ that arises from our criterion for $\equiv_r^k$-equivalence (Proposition 13). In order to more clearly

describe this strategy, we move over to the simpler setting of games on Kripke structures. In §6.1, we state and prove an analogous criterion for establishing $\sim_m$-equivalences among classes of Kripke structures with the same universe. In §6.2, we reprove this criterion in terms of games on Kripke structures to get a clear picture of the winning strategy that emerges. In Appendix B, we prove Proposition 13 from the analogous criterion on Kripke structures. In this way, we get a clear picture of the winning strategy in $\eth_r^k((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$.

For our purposes, we consider a restricted version of Kripke structures, namely colored directed graphs (digraphs) with a single distinguished vertex.

Definition 16. *Let $\mathcal{A} = (V, E)$ be a digraph. For $a \in V$, let $N_{\mathcal{A}}(a)$—or simply $N(a)$ if $\mathcal{A}$ is clear from context—denote the set of vertices $a' \in V$ such that $(a, a') \in E$. A function $f : V \longrightarrow \mathbb{N}$ is called a* coloring of $\mathcal{A}$. *The pair $(\mathcal{A}, f)$ is called a* colored digraph. *For $a \in V$, the triple $(\mathcal{A}, f, a)$ is called a* Kripke structure.

Definition 17. *For every $m \in \mathbb{N}$, equivalence relation $\approx_m$ on Kripke structures is defined inductively by*

- $(\mathcal{A}, f, a) \approx_0 (\mathcal{B}, g, b) \iff^{def} f(a) = g(b)$;
- *for $m \geqslant 1$, $(\mathcal{A}, f, a) \approx_m (\mathcal{B}, g, b) \iff^{def} f(a) = g(b)$ and*

$$\forall a' \in N_{\mathcal{A}}(a)\ \exists b' \in N_{\mathcal{B}}(b)\ (\mathcal{A}, f, a') \approx_{m-1} (\mathcal{B}, g, b'),$$
$$\forall b' \in N_{\mathcal{B}}(b)\ \exists a' \in N_{\mathcal{A}}(a)\ (\mathcal{A}, f, a') \approx_{m-1} (\mathcal{B}, g, b').$$

Remark 18. *The equivalence relation $\approx_m$ characterizes indistinguishability of Kripke structures up to sentences of rank m in* modal logic *(cp. Proposition 5).*

There is a simple characterization of $\approx_m$-equivalence in terms of appropriately defined games (cp. Definition 3).

Definition 19. *For Kripke structures $(\mathcal{A}, f, a)$ and $(\mathcal{B}, g, b)$ the game $\eth_m((\mathcal{A}, f, a), (\mathcal{B}, g, b))$ is defined as follows. There are two players, Spoiler and Duplicator. The "game board" consists of disjoint copies of digraphs $\mathcal{A}$ and $\mathcal{B}$. There are two "game pieces", α and β which sit on vertices of $\mathcal{A}$ and $\mathcal{B}$, respectively. Initially α sits on a and β sits on b. The game itself takes place in a sequence of m rounds. In each round, Spoiler moves first. Spoiler's move consists of choosing either $\mathcal{A}$ or $\mathcal{B}$ and moving the corresponding game piece forward along an edge. Duplicator then replies by moving the other game piece along an edge in the other graph. The "record" of a game is the sequence $(a_0, b_0), \ldots, (a_m, b_m)$ of positions before and after each of the m rounds (where $(a_0, b_0) = (a, b)$ is the initial position of α and β). Duplicator is declared the winner iff $f(a_i) = g(b_i)$ for all $i \in \{0, \ldots, m\}$. Duplicator is said to have a* winning strategy *if he can play in a manner such that he wins no matter what moves Spoiler makes.*

6.1 Criterion for $\approx_m$-Equivalence

Analogous to Proposition 13, we have the following criterion for $\approx_m$-equivalence.

Proposition 20. *Let $\mathcal{A} = (V, E)$ be a digraph, let $\{f_a\}_{a \in V}$ be a family of colorings $f_a : V \longrightarrow \mathbb{N}$ indexed by vertices a of $\mathcal{A}$, and let $g : V \longrightarrow \mathbb{N}$ be an additional coloring of $\mathcal{A}$. Suppose that*

i. *$f_a(a) = g(a)$ for all $a \in V$, and*
ii. *$(\mathcal{A}, f_a, b) \approx_m (\mathcal{A}, f_b, b)$ for all $(a, b) \in E$.*

Then $(\mathcal{A}, f_a, a) \approx_m (\mathcal{A}, g, a)$ for all $a \in V$.

Proof. Because we consider various Kripke structures with the same underlying graph, we simplify notation by abbreviating $(\mathcal{A}, f_a, b)$ and $(\mathcal{A}, g, a)$ as (f_a, b) and (g, a), respectively.

The proof is a simple argument by induction: for $\ell = 0, \ldots, m$ we claim that $(f_a, a) \approx_\ell (g, a)$ for every $a \in V$. The base case when $\ell = 0$ we get by hypothesis (i). For the induction step, assume that $\ell > 0$ and $(f_a, a) \approx_{\ell-1} (g, a)$ for every $a \in V$. Consider any $(a, b) \in E$. We have $(f_a, b) \approx_m (f_b, b)$ by hypothesis (ii). We have $(f_b, b) \approx_{\ell-1} (g, b)$ by the induction hypothesis. Therefore, we have $(f_a, b) \approx_{\ell-1} (g, b)$ by the fact that $\approx_m$ refines $\approx_{\ell-1}$ and by transitivity of $\approx_{\ell-1}$. We now check that

- $f_a(a) = g(a)$ (by hypothesis (i)),
- for all $a' \in N(a)$ there exists $\exists b' \in N(a)$ (namely $b' = a'$) such that $(f_a, a') \approx_{\ell-1} (g, b')$,
- for all $b' \in N(a)$ there exists $\exists a' \in N(a)$ (namely $a' = b'$) such that $(f_a, a') \approx_{\ell-1} (g, b')$.

Thus we have $(f_a, a) \approx_\ell (g, a)$ by definition of $\approx_\ell$.

6.2 Winning Strategy behind the $\approx_m$-Criterion (Proposition 20)

We now unravel the induction in the proof of Proposition 20 in order to extract a winning strategy in the game $\eth_m((f_a, a), (g, a))$. Suppose that for every edge $(a, b) \in E$, we are given a black-box winning strategy $\mathbf{S}(a, b)$ in the game $\eth_m((f_a, b), (f_b, b))$. Our goal is to devise a winning strategy—let's call it $\boldsymbol{\Sigma}(a)$—in the game $\eth_m((f_a, a), (g, a))$. Before describing the strategy $\boldsymbol{\Sigma}(a)$ in an informal manner, we prove a key lemma that captures the basic idea behind $\boldsymbol{\Sigma}(a)$.

Lemma 21. *Assume the hypotheses of Proposition 20. Let $\ell \in \{0, \ldots, m\}$, let $x_0, \ldots, x_\ell$ and $y_0, \ldots, y_\ell$ be vertices of $\mathcal{A}$ such that $x_\ell = y_\ell$ and*

$$(\dagger) \quad (f_{x_0}, y_0) \approx_{m-\ell} (f_{x_1}, y_1) \approx_{m-\ell+1} \cdots \approx_{m-1} (f_{x_{\ell-1}}, y_{\ell-1}) \approx_m (f_{x_\ell}, y_\ell).$$

Then

1. *$f_{x_0}(y_0) = g(y_\ell)$,*
2. *if $\ell < m$, then for every $z_0 \in N(y_0)$ there exists $(z_1, \ldots, z_\ell) \in N(y_1) \times \cdots \times N(y_\ell)$ such that*

$$(f_{x_0}, z_0) \approx_{m-\ell-1} (f_{x_1}, z_1) \approx_{m-\ell} \cdots \approx_{m-1} (f_{x_\ell}, z_\ell) \approx_m (f_{z_\ell}, z_\ell),$$

3. *if* $\ell < m$, *then for every* $z_\ell \in N(y_\ell)$ *there exists* $(z_0, \ldots, z_{\ell-1}) \in N(y_0) \times \cdots \times N(y_{\ell-1})$ *such that*

$$(f_{x_0}, z_0) \approx_{m-\ell-1} (f_{x_1}, z_1) \approx_{m-\ell} \cdots \approx_{m-1} (f_{x_\ell}, z_\ell) \approx_m (f_{z_\ell}, z_\ell).$$

Proof. We first prove statement (1). Hypothesis (†) implies that $(f_{x_0}, y_0) \approx_{m-\ell} (f_{x_\ell}, y_\ell)$. Therefore $f_{x_0}(y_0) = f_{x_\ell}(y_\ell)$. By hypothesis (i) of Proposition 20 we have $f_{y_\ell}(y_\ell) = g(y_\ell)$. Since $x_\ell = y_\ell$ we get $f_{x_0}(y_0) = g(y_\ell)$ as required.

For statement (2), assume $\ell < m$ and let $z_0 \in N(y_0)$. The fact that $(f_{x_0}, y_0) \approx_{m-\ell} (f_{x_1}, y_1)$ by (†) implies there exists $z_1 \in N(y_1)$ such that $(f_{x_0}, z_0) \approx_{m-\ell-1} (f_{x_1}, z_1)$. Next, the fact that $(f_{x_1}, y_1) \approx_{m-\ell+1} (f_{x_2}, y_2)$ by (†) implies there exists $z_2 \in N(y_2)$ such that $(f_{x_1}, z_1) \approx_{m-\ell} (f_{x_2}, z_2)$. Continuing in this fashion we obtain a sequence $(z_1, \ldots, z_\ell) \in N(y_1) \times \cdots \times N(y_\ell)$ such that

$$(f_{x_0}, z_0) \approx_{m-\ell-1} (f_{x_1}, z_1) \approx_{m-\ell} \cdots \approx_{m-2} (f_{x_{\ell-1}}, z_{\ell-1}) \approx_{m-1} (f_{x_\ell}, z_\ell).$$

Finally, we have $(f_{x_\ell}, z_\ell) \approx_m (f_{z_\ell}, z_\ell)$ by hypothesis (ii) of Proposition 20 since $z_\ell \in N(y_\ell) = N(x_\ell)$.

For statement (3), again assume $\ell < m$ and this time let $z_\ell \in N(y_\ell)$. From $(f_{x_{\ell-1}}, y_{\ell-1}) \approx_m (f_{x_\ell}, y_\ell)$, it follows that there exists $z_{\ell-1} \in N(y_{\ell-1})$ such that $(f_{x_{\ell-1}}, z_{\ell-1}) \approx_{m-1} (f_{x_\ell}, z_\ell)$. Continuing, we get a sequence $(z_0, \ldots, z_{\ell-1}) \in N(y_0) \times \cdots \times N(y_{\ell-1})$ such that

$$(f_{x_0}, z_0) \approx_{m-\ell-1} (f_{x_1}, z_1) \approx_{m-\ell} \cdots \approx_{m-2} (f_{x_{\ell-1}}, z_{\ell-1}) \approx_{m-1} (f_{x_\ell}, z_\ell).$$

As before, we get $(f_{x_\ell}, z_\ell) \approx_m (f_{z_\ell}, z_\ell)$ by Proposition 20(ii) since $z_\ell \in N(y_\ell) = N(x_\ell)$.

Lemma 21 implicitly contains a winning strategy $\boldsymbol{\Sigma}(a)$ for Duplicator in the game $\Im_m((f_a, a), (g, a))$ for every $a \in V$, given a family $\{\mathbf{S}(u, v)\}_{(u,v)\in E}$ of black-box winning strategies in games $\Im_m((f_u, v), (f_v, v))$ for all edges $(u, v) \in E$. Suppose $(a_0, b_0), \ldots, (a_\ell, b_\ell)$ is the sequence of positions before and after each of the first ℓ rounds of $\Im_m((f_a, a), (g, a))$. (In particular, $(a_0, b_0) = (a, a)$ is the initial position.) Under the strategy $\boldsymbol{\Sigma}(a)$, Duplicator maintains sequences $x_0, \ldots, x_\ell$ and $y_0^{(\ell)}, \ldots, y_\ell^{(\ell)}$ such that

- $x_0 = a$ and $y_0^{(\ell)} = a_\ell$ and $x_\ell = y_\ell^{(\ell)} = b_\ell$,
- $x_0, \ldots, x_\ell$ is a path in A, and
- (†) holds, that is,

$$(f_{x_0}, y_0^{(\ell)}) \approx_{m-\ell} (f_{x_1}, y_1^{(\ell)}) \approx_{m-\ell+1} \cdots \approx_{m-1} (f_{x_{\ell-1}}, y_{\ell-1}^{(\ell)}) \approx_m (f_{x_\ell}, y_\ell^{(\ell)}).$$

Assuming $\ell < m$, Duplicator plays under strategy $\boldsymbol{\Sigma}(a)$ in round $\ell + 1$ as follows. First, suppose Spoiler plays $a_{\ell+1} \in N(a_\ell)$ in the colored digraph $(\mathcal{A}, f_a)$. Then Duplicator sets $z_0 = a_{\ell+1}$, using strategies $\mathbf{S}(x_0, x_1), \ldots, \mathbf{S}(x_{\ell-1}, x_\ell)$ (note that $(x_{i-1}, x_i) \in E$ for all $1 \leqslant i \leqslant \ell$), constructs the sequence $z_1, \ldots, z_\ell$ as in Lemma 21(2). Duplicator then plays $z_\ell \in N(b_\ell)$ in $(\mathcal{A}, g)$ and updates his internal bookkeeping by setting $x_{\ell+1} = z_\ell$ and $y_0^{(\ell+1)} = z_0, \ldots, y_\ell^{(\ell+1)} = z_\ell$ and $y_{\ell+1}^{(\ell+1)} = z_\ell$. If instead Spoiler plays $b_{\ell+1} \in N(b_\ell)$ in $(\mathcal{A}, g)$, then Duplicator responds in a similar manner using Lemma 21(3).

References

[1] Ajtai, M.: Σ_1^1 formulae on finite structures. Annals of Pure and Applied Logic 24, 1–48 (1983)

[2] Amano, K., Maruoka, A.: A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6)\log\log n$ negation gates. SIAM J. Comput. 35(1), 201–215 (2005)

[3] Beame, P.: Lower bounds for recognizing small cliques on CRCW PRAM's. Discrete Appl. Math. 29(1), 3–20 (1990)

[4] Beame, P.: A switching lemma primer. Technical Report UW-CSE-95-07-01, Department of Computer Science and Engineering, University of Washington (November 1994)

[5] Boppana, R.B.: The average sensitivity of bounded-depth circuits. Inf. Process. Lett. 63(5), 257–261 (1997)

[6] Dawar, A.: How many first-order variables are needed on finite ordered structures? In: We Will Show Them: Essays in Honour of Dov Gabbay, pp. 489–520 (2005)

[7] Denenberg, L., Gurevich, Y., Shelah, S.: Definability by constant-depth polynomial-size circuits. Information and Control 70(2/3), 216–240 (1986)

[8] Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Mathematical Systems Theory 17, 13–27 (1984)

[9] Goldmann, M., Håstad, J.: A simple lower bound for the depth of monotone circuits computing clique using a communication game. Information Processing Letters 41(4), 221–226 (1992)

[10] Gurevich, Y., Lewis, H.R.: A logic for constant-depth circuits. Information and Control 61(1), 65–74 (1984)

[11] Håstad, J.: Almost optimal lower bounds for small depth circuits. In: STOC 1986: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, pp. 6–20 (1986)

[12] Immerman, N.: Upper and lower bounds for first order expressibility. J. Comput. Syst. Sci. 25(1), 76–98 (1982)

[13] Immerman, N.: Descriptive Complexity. In: Graduate Texts in Computer Science. Springer, New York (1999)

[14] Immerman, N., Buss, J., Barrington, D.M.: Number of variables is equivalent to space. Journal of Symbolic Logic 66 (2001)

[15] Koucky, M., Lautemann, C., Poloczek, S., Therien, D.: Circuit lower bounds via Ehrenfeucht-Fraisse games. In: CCC 2006: Proceedings of the 21st Annual IEEE Conference on Computational Complexity, pp. 190–201 (2006)

[16] Libkin, L.: Elements of Finite Model Theory. Springer, Heidelberg (2004)

[17] Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, fourier transform, and learnability. J. ACM 40(3), 607–620 (1993)

[18] Lynch, J.F.: A depth-size tradeoff for boolean circuits with unbounded fan-in. In: Structure in Complexity Theory Conference, pp. 234–248 (1986)

[19] Razborov, A.A.: Lower bounds on the size of bounded depth networks over a complete basis with logical addition. Matematicheskie Zametki 41, 598–607 (1987); English translation in Mathematical Notes of the Academy of Sciences of the USSR 41, 333–338 (1987) (in Russian)

[20] Rossman, B.: On the constant-depth complexity of k-clique. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 721–730 (2008)

[21] Smolensky, R.: Algebraic methods in the theory of lower bounds for boolean circuit complexity. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, pp. 77–82 (1987)

A Proof of Theorem 8

We begin by defining circuits, the complexity class $\mathrm{AC}^0(\text{depth } d)$ and the average sensitivity of functions with domain $\{0,1\}^n$.

Definition 22. *A* circuit on n inputs *is an acyclic directed graph $\mathscr{C}$ in which sources are labeled by elements of $[n] \times \{+,-\}$ (where $(i,+)$ corresponds to the literal x_i and $(i,-)$ corresponds to the literal $\overline{x_i}$) and all other nodes are labeled by $\wedge$ or $\vee$. Sinks in $\mathscr{C}$ are called* outputs. *The circuit $\mathscr{C}$ computes a Boolean function $\{0,1\}^n \longrightarrow \{0,1\}^m$ where $m = |\{\text{output nodes in } \mathscr{C}\}|$.*

Definition 23. *Let $\bar{\mathscr{C}} = (\mathscr{C}_n)_{n\in\mathbb{N}}$ be a sequence of circuits $\mathscr{C}_n$ on n inputs. We say that $\bar{\mathscr{C}} \in \mathrm{AC}^0(\text{depth } d)$ if*

- *$\mathscr{C}_n$ has size $n^{O(1)}$ and*
- *there exists a constant c such that for all n, every directed path in $\mathscr{C}_n$ contains at most d nodes of fan-in (i.e., in-degree) $> c$.*

Definition 24. *For a function f with domain $\{0,1\}^n$ and an element $x \in \{0,1\}^n$, the* sensitivity of f at x *is defined by*

$$\mathrm{sens}(f,x) \triangleq |\{i \in [n] : f(x) \neq f(x_1,\ldots,x_{i-1},1-x_i,x_{i+1},\ldots,x_n)\}|.$$

The average sensitivity of f *is defined by*

$$\text{ave-sens}(f) \triangleq 2^{-n} \textstyle\sum_{x\in\{0,1\}^n} \mathrm{sens}(f,x).$$

The next lemma is a fundamental result in circuit complexity.

Lemma 25 ([17, 5]). *Suppose $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ is computed by a circuit in $\mathrm{AC}^0(\text{depth } d)$. Then* $\text{ave-sens}(f) = O\left(m \log^{d-1} n\right)$.

We now make an observation about strong r-equivalence akin to well-known results in descriptive complexity characterizing first-order logic in terms of AC^0 (see [7, 10, 12, 13]).

Lemma 26. *Let $\mathcal{A}$ be a structure with universe $[n]$ (really, a sequence of structures, one for each natural number n). There exists a function (really, a sequence of functions) $f_{\mathcal{A}} : \{0,1\}^n \longrightarrow \{0,1\}^{O(\log n)}$ computed by circuits in $\mathrm{AC}^0(\text{depth } O(r))$ such that for all $P,Q \subseteq [n]$, structures $(\mathcal{A},P)$ and $(\mathcal{A},Q)$ are strongly r-equivalent iff $f_{\mathcal{A}}(P) = f_{\mathcal{A}}(Q)$ (where we view P and Q as elements of $\{0,1\}^n$).*

That is, $f_{\mathcal{A}}(P)$ completely describes the set of $(\mathcal{A},P)$-minimal sequences $(a_1,\ldots,a_r)$ of length r as well as substructures $(\mathcal{A},P)|_{\{a_1,\ldots,a_r\}}$.

Proof (sketch). We use the fact that the number of minimal sequences of length r in any structure is bounded by an absolute constant (by Lemma 10). We can thus represent the set of all $(\mathcal{A},P)$-minimal sequences $(a_1,\ldots,a_r)$ of length r as well as substructures $(\mathcal{A},P)|_{\{a_1,\ldots,a_r\}}$ using only $O(\log n)$ bits. This can be achieved by a polynomial-size circuit of depth $O(r)$ using standard arguments (cp. [7, 10, 12, 13]).

Using Lemmas 25 and 26, we easily prove Theorem 4. As remarked at the end of §3, it suffices to prove

$$\Pr\big[(\mathcal{A},P) \text{ and } (\mathcal{A},P') \text{ are not strongly } r\text{-equivalent}\big] \leqslant O\big((\log n)^{O(r)}/n\big)$$

where $\mathcal{A}$ is an arbitrary structure with universe $[n]$, P is a uniform random subset of $[n]$, and $P' = P \vartriangle \{q\}$ where q is a uniform random element of $[n]$. But the statement that $(\mathcal{A},P)$ and $(\mathcal{A},P')$ are not strongly r-equivalent is equivalent to $f_{\mathcal{A}}(P) \neq f_{\mathcal{A}}(P')$. Completing the proof, we have

$$\Pr[f_{\mathcal{A}}(P) \neq f_{\mathcal{A}}(P')] = \mathrm{E}\,[\mathrm{sens}(f,P)/n] = \text{ave-sens}(f)/n = O\big((\log n)^{O(r)}/n\big).$$

B Proof of Proposition 13

In order to prove Proposition 13 (the criterion for $\equiv_r^k$-equivalence), we first state a suitable generalization of Proposition 20 (the criterion for $\approx_r$-equivalence of Kripke structures). Rather than digraphs, we consider *k-digraphs* $\mathcal{A} = (V, E_1, \ldots, E_k)$ where each E_i is a binary relation on V. For $a \in V$ and $i \in [k]$, let $N_{\mathcal{A},i}(a)$ for the neighbor set of a under edge relation E_i and let $N_{\mathcal{A}}(a) = N_{\mathcal{A},1}(a) \cup \cdots \cup N_{\mathcal{A},k}(a)$.

A *k-Kripke structure* is a triple $(\mathcal{A}, f, a)$ where $\mathcal{A}$ is a k-digraph and $f : V \longrightarrow \mathbb{N}$ is a coloring of A and $a \in V$ is a distinguished vertex. $\approx_r$-equivalence of k-Kripke structures is defined as follows (cf. Definition 2 of $\equiv_r^k$-equivalence):

- $(\mathcal{A}, f, a) \approx_0 (\mathcal{B}, g, b) \iff^{\text{def}} f(a) = g(b)$;
- for $r \geqslant 1$, $(\mathcal{A}, f, a) \approx_r (\mathcal{B}, g, b) \iff^{\text{def}} f(a) = g(b)$ and for all $i \in [k]$,

$$\forall a' \in N_{\mathcal{A},i}(a)\ \exists b' \in N_{\mathcal{B},i}(b)\ (\mathcal{A}, f, a') \approx_{r-1} (\mathcal{B}, g, b'),$$
$$\forall b' \in N_{\mathcal{B},i}(b)\ \exists a' \in N_{\mathcal{A},i}(a)\ (\mathcal{A}, f, a') \approx_{r-1} (\mathcal{B}, g, b').$$

We have the following generalization of Proposition 20 to k-Kripke structures.

Proposition 27. *Let $\mathcal{A} = (V, E_1, \ldots, E_k)$ be a k-digraph, let $\{f_a\}_{a\in V}$ be a family of colorings $f_a : V \longrightarrow \mathbb{N}$ indexed by vertices $a \in V$, and let $g : V \longrightarrow \mathbb{N}$ be another coloring of A. Suppose that*

i'. *$f_a(a) = g(a)$ for all $a \in V$, and*
ii'. *$(\mathcal{A}, f_a, b) \approx_r (\mathcal{A}, f_b, b)$ for all $(a,b) \in E_1 \cup \cdots \cup E_k$.*

Then $(\mathcal{A}, f_a, a) \approx_r (\mathcal{A}, g, a)$ for all $a \in V$.

The proof of Proposition 27 is virtually identical to the proof of Proposition 20 in §6.1. The game version of this proof given in §6.2 likewise generalizes to the k-Kripke setting.

We are ready to derive Proposition 13 from Proposition 27. Let $V, \sim, \{\mathcal{G}_{\bar{v}}\}_{\bar{v}\in V^k}$ and $\mathcal{G}^*$ be as in Proposition 13. Assume that hypotheses (i) and (ii) hold:

i. $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_0 (\mathcal{G}^*, \bar{v})$ for all $\bar{v} \in V^k$, and
ii. $(\mathcal{G}_{\bar{v}}, \bar{v}) \equiv_{rd} (\mathcal{G}_{\bar{w}}, \bar{v})$ for all $\bar{v}, \bar{w} \in V^k$ such that $\bar{v} \sim \bar{w}$.

Consider the k-digraph $\mathcal{A} = (V^k, E_1, \ldots, E_k)$ with vertex set V^k and edge relations

$$E_i = \big\{(\bar{v}, \bar{w}) \in V^k \times V^k : v_i \sim w_i \text{ and } v_j = w_j \text{ for all } j \in [k] \setminus \{i\}\big\}.$$

Fix an arbitrary enumeration of the (finitely many) $\equiv_0$-equivalence classes of k-tupled structures. For all $\bar{v} \in V^k$, define $f_{\bar{v}} : V^k \longrightarrow \mathbb{N}$, and also define $g : V^k \longrightarrow \mathbb{N}$, by

$$f_{\bar{v}}(\bar{w}) = \text{index of the } \equiv_0\text{-equivalence class of } (\mathcal{G}_{\bar{v}}, \bar{w}),$$
$$g(\bar{v}) = \text{index of the } \equiv_0\text{-equivalence class of } (\mathcal{G}^*, \bar{v}).$$

Check that

- $f_{\bar{v}}(\bar{v}) = g(\bar{v})$ for all $\bar{v} \in V^k$ (by hypothesis (i)),
- $f_{\bar{v}}(\bar{w}) \approx_{rd} f_{\bar{w}}(\bar{w})$ for all $(\bar{v}, \bar{w}) \in E_1 \cup \cdots \cup E_k$ (using hypothesis (ii)).

Therefore, by Proposition 27 we have $(\mathcal{A}, f_{\bar{v}}, \bar{v}) \approx_{rd} (\mathcal{A}, g, \bar{v})$ for all $\bar{v} \in V^k$.

But what exactly does $(\mathcal{A}, f_{\bar{v}}, \bar{v}) \approx_{rd} (\mathcal{A}, g, \bar{v})$ mean? It is easy to see that this is equivalent to the existence of a winning strategy in the following $\sim$-*constrained* rd-round k-pebble game, which we denote $\widetilde{\eth}^k_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$: the $\sim$-constrained game is just like the usual rd-round k-pebble game, except that when either player moves a pebble from its present location on an element v, he is required to place that pebble on an element w such that $v \sim w$. To complete the proof of Proposition 13, we claim that a winning strategy in $\widetilde{\eth}^k_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$ implies a winning strategy in the usual r-round k-pebble game $\eth^k_r((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$. Here we exploit the fact that $\sim$ has diameter $\leqslant d$, which lets us map each move in $\eth^k_r((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$ to a sequence of $\leqslant d$ moves in $\widetilde{\eth}^k_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$. We thus get a winning strategy by playing d moves in a simulation of $\widetilde{\eth}^k_{rd}((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$ for each move in the game $\eth^k_r((\mathcal{G}_{\bar{v}}, \bar{v}), (\mathcal{G}^*, \bar{v}))$.

Sound and Complete Tree-Sequent Calculus for Inquisitive Logic

Katsuhiko Sano*

Department of Humanistic Informatics
Graduate School of Letters
Kyoto University / JSPS
katsuhiko.sano@gmail.com

Abstract. We introduce a tree-sequent calculus for inquisitive logic (Groenendijk 2008) as a special form of labelled deductive system (Gabbay 1996). In particular, we establish that (i) our tree-sequent calculus is sound and complete with respect to Groenendijk's inquisitive semantics and that (ii) our tree-sequent calculus is decidable and enjoys cut-elimination theorem. (ii) is semantically revealed by our argument for (i). The key idea which allows us to obtain these results is as follows: In Groenendijk's inquisitive semantics, a formula of propositional logic is evaluated against a pair of worlds on a model. Given the appropriate pre-order on the set of such pairs, any inquisitive model can be regarded as a Kripke model for intuitionistic logic. This representation enables us to connect inquisitive semantics with the tree-sequent technique for non-classical logics (Kashima 1999).

1 Introduction

Jeroen Groenendijk [8] introduced the *inquisitive semantics* for a language of propositional logic. Unlike his earlier work, *Logic of Interrogation* [7], the syntactic distinction between the categories of declarative and interrogative sentences is not carried over to the semantics. That is, often, questions are uninformative and assertions are uninquisitive. In the current setting, however, some sentences can be both informative and inquisitive in evaluating a formula against a pair of worlds. For example, a plain contingent disjunction $p \vee q$ is counted as such.

Groenendijk [8] gave only a semantic definition of his logic for inquisitive semantics. As suggested in [3, p.63], however, we also need to understand the proof theory of this logic for practical applications like building question-answering systems. This paper contributes to this point. That is, we give a sound and complete Gentzen-style sequent calculus to strengthen the logical basis for inquisitive semantics.

We study the inquisitive logic employing the method of tree-sequent calculus [12], which is a special form of Labelled Deductive System [6]. This style of formulation is

* I would like to thank Yurie Hara for introducing me to the inquisitive semantics in D. Lewis reading group at Kyoto University and checking my English. I also would like to thank Floris Roelofsen for his comments on the earlier version of this paper. I wish to thank Jeroen Groenendijk for connecting me with him. I would like to thank the anonymous referees for their comments and suggestions. Finally, I am grateful to Ryo Kashima for introducing me to the tree-sequent technique in May 2004. All errors, however, are mine.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 365–378, 2009.

useful in axiomatizing various logics defined through Kripke models without regard to whether they are expressed in intuitionistic or modal languages (see, e.g., [9,10,11]). Now, the question pertains to: how we shall bridge the gap between the inquisitive semantics and the method of tree-sequent calculus? The answer is to represent a model of inquisitive semantics as a Kripke model for intuitionistic logic.

Recently, Salvador Mascarenhas [13] has provided a complete Hilbert-style axiomatization for inquisitive logic. Independently, Ivano Ciardelli and Floris Roelofsen [5] also have established the completeness results for a hierarchy of inquisitive logics, one of which corresponds to our tree-sequent calculus. Unlike our approach, both of these works have based on Hilbert-style axiomatization. Since our tree-sequent calculus can be converted into a tableau calculus, it is easier to show a theorem or construct a counterexample. Furthermore, we show that our tree-sequent calculus is decidable and enjoys cut-elimination theorem.

The organization of the paper is as follows. In Section 2, we define inquisitive semantics and extract the semantical essence by examining the relation between inquisitive semantics and Kripke semantics for intuitionistic logic. In Section 3, we introduce our tree-sequent calculus for inquisitive logic and give some examples of its derivations. In Section 4, we prove the completeness of the (cut-free) tree-sequent calculus by constructing saturated tree-sequents (see Theorem 1). In Section 5, we introduce a translation of tree-sequents into formulas, and then establish the soundness of our tree-sequent calculus (with cut-rule) (see Theorem 2). As a corollary to our completeness and soundness results, Section 6 shows that our tree-sequent calculus is decidable and enjoys cut-elimination theorem (see Corollary 1).

2 Inquisitive Semantics

First of all, let us introduce our syntax. Our language $\mathcal{L}_{\wp}$ consists of (i) a countable set of propositional variables $\wp$; (ii) the propositional connectives: $\bot$, $\neg$, $\rightarrow$, $\wedge$ and $\vee$; and (iii) the parentheses: (,). Then, the formulas of $\mathcal{L}_{\wp}$ are defined inductively by:

$$A ::= p \mid \bot \mid (\neg A) \mid (A \wedge B) \mid (A \vee B) \mid (A \rightarrow B).$$

We define $\top := \neg\bot$, $!A := \neg\neg A$, $?A := A \vee \neg A$. For a finite set Γ of formulas, $\bigwedge \Gamma$ (or, $\bigvee \Gamma$) is defined as the conjunction (or, disjunction) of all formulas of Γ, if Γ is non-empty; otherwise $\top$ (or, $\bot$, respectively). Groenendijk [8] referred to $!A$ as an *assertion*, and to $?A$ as a *question*.

An *inquisitive model* $\mathfrak{M}$ is a pair consisting of (i) a non-empty set W, called the *domain* of $\mathfrak{M}$, and (ii) a valuation $V : \wp \rightarrow \mathcal{P}(W)$. Given any inquisitive model $\mathfrak{M} = \langle W, V \rangle$, any $w, v \in W$ and any formula A, the satisfaction relation $(w, v) \models_{\mathfrak{M}} A$ is defined by:

$$\begin{aligned}
(w, v) &\models_{\mathfrak{M}} p \text{ iff } w \in V(p) \text{ and } v \in V(p);\\
(w, v) &\models_{\mathfrak{M}} \bot \text{ iff Never};\\
(w, v) &\models_{\mathfrak{M}} \neg A \text{ iff for all pairs } \pi \text{ in } \{ w, v \}\text{: } \pi \not\models_{\mathfrak{M}} A;
\end{aligned}$$

$$(w,v) \models_{\mathfrak{M}} A \wedge B \text{ iff } (w,v) \models_{\mathfrak{M}} A \text{ and } (w,v) \models_{\mathfrak{M}} B;$$
$$(w,v) \models_{\mathfrak{M}} A \vee B \text{ iff } (w,v) \models_{\mathfrak{M}} A \text{ or } (w,v) \models_{\mathfrak{M}} B;$$
$$(w,v) \models_{\mathfrak{M}} A \to B \text{ iff for all pairs } \pi \text{ in } \{w,v\}\text{: } \pi \models_{\mathfrak{M}} A \text{ implies } \pi \models_{\mathfrak{M}} B.$$

We usually drop the subscript $\mathfrak{M}$ from $\models_{\mathfrak{M}}$, if it is clear from the context.

We will use the following [8, Theorem 2 and Proposition 2].

Proposition 1. *Let w, v in $\mathfrak{M}$.* (i) $(w,v) \models A$ *implies* $(v,w) \models A$. (ii) $(w,v) \models A$ *implies* $(w,w) \models A$ *and* $(v,v) \models A$. (iii) $(w,v) \models \neg A$ iff $(w,w) \not\models A$ *and* $(v,v) \not\models A$.

By (i), it suffices to consider the pairs (w,w), (v,v), (w,v) from $\{w,v\}$ for the conditions of satisfaction of $\neg A$ and $A \to B$. By (iii), we can spell out the conditions of satisfaction of $!A$ and $?A$ as follows:

$$(w,v) \models !A \text{ iff } (w,w) \models A \text{ and } (v,v) \models A,$$
$$(w,v) \models ?A \text{ iff } (w,v) \models A \text{ or } ((w,w) \not\models A \text{ or } (v,v) \not\models A).$$

The following proposition tells us that any diagonal pairs (w,w) behave classically.

Proposition 2. *Given any $\mathfrak{M}$ and any w in $\mathfrak{M}$,* (i) $(w,w) \models \neg A$ iff $(w,w) \not\models A$; (ii) $(w,w) \models A \to B$ iff $(w,w) \models A$ *implies* $(w,w) \models B$.

Given any $\mathfrak{M} = \langle W, V \rangle$, A is *valid in* $\mathfrak{M}$ (notation: $\models_{\mathfrak{M}} A$) if for any $w, v \in W$, $(w,v) \models_{\mathfrak{M}} A$. Let M be a class of inquisitive models. We say that A is *valid in* M if $\models_{\mathfrak{M}} A$ for any inquisitive model $\mathfrak{M}$ in M.

Definition 1. $\mathsf{M}_{\mathrm{all}}$ *is the class of all inquisitive models and* $\mathsf{M}_2 := \{\langle W, V \rangle \mid \#W = 2\}$. $\mathsf{M}_1 := \{\langle W, V \rangle \mid \#W = 1\}$ *and* $\mathsf{M}_{\geq 2} := \{\langle W, V \rangle \mid \#W \geq 2\}$.

Proposition 3. *Assume that $\#W \geq 2$. Then, A is valid in an inquisitive model $\mathfrak{M}$* iff $(w,v) \models A$ *for any distinct w, v in $\mathfrak{M}$.*

Proof. By Proposition 1 (ii). □

Proposition 4. (i) *A is valid in M_1* iff *A is a truth-functional tautology.* (ii) *If A is valid in $\mathsf{M}_{\geq 2}$, then A is a truth-functional tautology.*

Proof. (i) is trivial from Proposition 2. Let us show the contrapositive implication of (ii). Assume that A is not a truth-functional tautology. So, there exists some truth-functional valuation $\mathcal{V} : \wp \to \{\mathbf{t}, \mathbf{f}\}$ such that $\mathcal{V}(A) \neq \mathbf{t}$. Define $\mathfrak{M} = \langle W, V \rangle$ as follows: $W := \{0,1\}$, $V(p) := W$ (if $\mathcal{V}(p) = \mathbf{t}$); $\emptyset$ (o.w.). Then, we can easily show that: for any B, $(0,0) \models_{\mathfrak{M}} B$ iff $\mathcal{V}(B) = \mathbf{t}$. Thus, we conclude $(0,0) \not\models_{\mathfrak{M}} A$. □

By Proposition 4, the validity in $\mathsf{M}_{\geq 2}$ implies the validity in M_1. So, whenever we consider the validity in $\mathsf{M}_{\mathrm{all}}$, we need not consider M_1. By Propositions 3 and 4, we deduce the following.

Proposition 5. *A is valid in $\mathsf{M}_{\mathrm{all}}$* iff $(w,v) \models_{\langle W,V \rangle} A$ *for any distinct w, v in W and any $\langle W, V \rangle$ in $\mathsf{M}_{\geq 2}$.*

Some readers may find that inquisitive semantics have some similarity to *Kripke semantics for intuitionistic logic*. By the following procedure, we can regard any inquisitive model $\mathfrak{M} = \langle W, V \rangle$ as a Kripke model for intuitionistic logic. Let us define the binary relation $\leq$ on $W \times W$ by: $(w, v) \leq (w', v')$ iff (w', v') is a pair from $\{ w, v \}$. Clearly $\leq$ is a reflexive and transitive relation. Next, we define $V' : \wp \to \mathcal{P}(W \times W)$ by $V'(p) := V(p) \times V(p)$. Then, $V'(p)$ satisfies the hereditary condition, i.e., $(w, v) \in V'(p)$ and $(w, v) \leq (w', v')$ implies $(w', v') \in V'(p)$. Thus, $\langle W \times W, \leq, V' \rangle$ is a Kripke model for intuitionistic logic. Let $\Vdash$ be the satisfaction relation of Kripke semantics for intuitionistic logic. Then, we can show that $(w, v) \models_{\mathfrak{M}} A$ iff $\langle W \times W, \leq, V' \rangle, (w, v) \Vdash A$ for any $w, v \in W$ and any formula A. Therefore:

$$(w, v) \models A \to B \text{ iff } \text{ for any } (w', v') \in W \times W\text{: } (w, v) \leq (w', v') \text{ implies } (w', v') \models A.$$

This observation allows us to say that all theorems of intuitionistic logic are valid in $\mathsf{M}_{\mathrm{all}}$ (cf. [14, Section 3]).

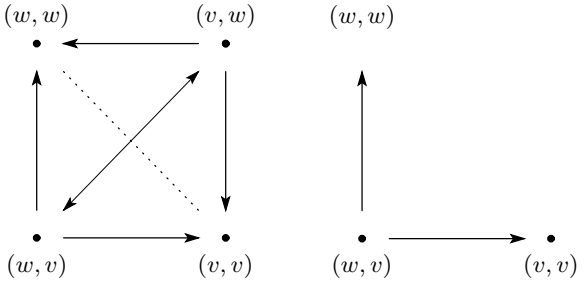


Fig. 1. How to extract the semantical basis for our tree-sequent

The above representation of a inquisitive model as a Kripke model also gives us the semantical basis of our tree-sequent calculus introduced in Section 3 as follows. Proposition 5 tells us that any distinct pair is crucial for validity. Let us fix such a distinct pair (w, v). In order to evaluate any formula at (w, v), it suffices to consider all the $\leq$-connected pairs, where $\leq$ is defined as above (rigorously speaking, we need the notion of *generated submodel* [4, p.29] here). Then we obtain the situation depicted in the left part of Figure 1, where the solid lines represent the $\leq$-relation (but we omit all the reflexive arrows here). Proposition 1 (i) allows us to fold this 'square' on the dotted diagonal line and to obtain the tree-structure depicted in the right part of Figure 1. This is the semantical basis for our tree-sequent calculus. We will label this tree with 0, 1 and 2 in Section 3 (see Definition 2).

3 Tree-Sequent Calculus for Inquisitive Logic

In this section, we introduce a tree-sequent calculus for *Inquisitive Logic*, i.e., the set of formulas valid in $\mathsf{M}_{\mathrm{all}}$, as a special form of Labelled Deductive Systems [6].

Definition 2 (tree-sequent). *Let* $\mathcal{T} = \langle \{ 0, 1, 2 \}, \leq \rangle$ *be the tree equipped with the order* $\leq := \{ \langle 0, 1 \rangle, \langle 0, 2 \rangle \} \cup \{ \langle x, x \rangle \mid x \in \{ 0, 1, 2 \} \}$. *A label is an element of*

$\{0, 1, 2\}$. *We use letters* α, β, *etc. for labels. A labelled formula is a pair* $\alpha : A$, *where* α *is a label and* A *is a formula of the language* $\mathcal{L}_{\wp}$. *A tree-sequent is an expression* $\Gamma \Rightarrow \Delta$ *where* Γ *and* Δ *are finite sets of labelled formulas.*

Unlike the previous studies [9,10,11], we use the *fixed* tree $\mathcal{T}$ here. This difference become important when we prove the soundness in Section 5.

Now, let us introduce the tree-sequent calculus $\mathbf{T}$InqL for inquisitive logic. This system defines inference schemes which allow us to manipulate tree-sequents. The axioms (i.e., the initial tree-sequents) of $\mathbf{T}$InqL are of the following forms:

$$\alpha : A, \Gamma \Rightarrow \Delta, \alpha : A \ (\text{Ax}) \qquad \alpha : \bot, \Gamma \Rightarrow \Delta \ (\bot\text{L}).$$

The inference rules of $\mathbf{T}$InqL are the following:

$$\frac{0 : p, \Gamma \Rightarrow \Delta}{1 : p, 2 : p, \Gamma \Rightarrow \Delta} \ (\text{Atom L}) \qquad \frac{1 : A, 2 : A, \Gamma \Rightarrow \Delta}{0 : A, \Gamma \Rightarrow \Delta} \ (\text{Move})$$

$$\frac{\alpha : A, \alpha : B, \Gamma \Rightarrow \Delta}{\alpha : A \wedge B, \Gamma \Rightarrow \Delta} \ (\wedge\text{L}) \qquad \frac{\Gamma \Rightarrow \Delta, \alpha : A \quad \Gamma \Rightarrow \Delta, \alpha : B}{\Gamma \Rightarrow \Delta, \alpha : A \wedge B} \ (\wedge\text{R})$$

$$\frac{\alpha : A, \Gamma \Rightarrow \Delta \quad \alpha : B, \Gamma \Rightarrow \Delta}{\alpha : A \vee B, \Gamma \Rightarrow \Delta} \ (\vee\text{L}) \qquad \frac{\Gamma \Rightarrow \Delta, \alpha : A, \alpha : B}{\Gamma \Rightarrow \Delta, \alpha : A \vee B} \ (\vee\text{R})$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha : A}{\alpha : \neg A, \Gamma \Rightarrow \Delta} \ (\neg\text{L}) \qquad \frac{\alpha : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \alpha : \neg A} \ (\neg\text{R}_{1,2}) \text{ where } \alpha \neq 0$$

$$\frac{1 : A, \Gamma \Rightarrow \Delta \quad 2 : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, 0 : \neg A} \ (\neg\text{R}_0)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha : A \quad \alpha : B, \Gamma \Rightarrow \Delta}{\alpha : A \to B, \Gamma \Rightarrow \Delta} \ (\to\text{L}) \qquad \frac{\alpha : A, \Gamma \Rightarrow \Delta, \alpha : B}{\Gamma \Rightarrow \Delta, \alpha : A \to B} \ (\to\text{R}_{1,2}) \text{ where } \alpha \neq 0$$

$$\frac{0 : A, \Gamma \Rightarrow \Delta, 0 : B \quad 1 : A, \Gamma \Rightarrow \Delta, 1 : B \quad 2 : A, \Gamma \Rightarrow \Delta, 2 : B}{\Gamma \Rightarrow \Delta, 0 : A \to B} \ (\to\text{R}_0)$$

$$\frac{\Gamma \Rightarrow \Delta, \alpha : A \quad \alpha : A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \ (\text{Cut})$$

The tree-sequent calculus $\mathbf{cutfree}\mathbf{T}$InqL is obtained by dropping (Cut) from $\mathbf{T}$InqL. Whenever a tree-sequent $\Gamma \Rightarrow \Delta$ is provable in $\mathbf{T}$InqL (or, in $\mathbf{cutfree}\mathbf{T}$InqL), we write $\mathbf{T}\mathsf{InqL} \vdash \Gamma \Rightarrow \Delta$ (or, $\mathbf{cutfree}\mathbf{T}\mathsf{InqL} \vdash \Gamma \Rightarrow \Delta$, respectively).

Remark 1. We can remove either $\bot$ or $\neg$ from our vocabulary. As for $\bot$, it suffices to define $\bot := p \wedge \neg p$. Then we can easily derive ($\bot$ L) from (Ax) and ($\wedge$L). Let us describe how to eliminate $\neg$. First, in our system, notice that the (depth-preserving) weakening is admissible, i.e., if one can derive $\Gamma \Rightarrow \Delta$, then one can also derive $\alpha : A, \Gamma \Rightarrow \Delta, \beta : B$, where α and β are any label from 0, 1, 2. Then, in order to establish our aim, it suffices to show the rules about the negation are derivable in the system without the negation. ($\neg$L) is easy to show. As for ($\neg\text{R}_{1,2}$) and ($\neg\text{R}_0$), it suffices to use the weakening above (remark that we also need (Move) when deriving ($\neg\text{R}_0$)).

Let us give some derivations in our tree-sequent calculus. We say that A of $\mathcal{L}_\wp$ is *provable* in $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$ (or, $\mathbf{T}\mathsf{InqL}$) if $\Rightarrow 0 : A$ is provable in $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$ (or, $\mathbf{T}\mathsf{InqL}$, respectively).

Proposition 6. *The following are all provable in* $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$*:*

(i) $\neg\neg p \to p$
(ii) $(\neg A \to B \vee C) \to (\neg A \to B) \vee (\neg A \to C)$
(iii) $(\neg A \to ?B) \to (\neg A \to B) \vee (\neg A \to \neg B)$
(iv) $A \vee (A \to (B \vee \neg B))$
(v) $(A \to (B \vee C)) \vee (B \to (A \vee C)) \vee (C \to (A \vee B))$

Let us explain these formulas. As for (i), remark that $\neg\neg A \to A$ is not valid in $\mathsf{M}_{\mathrm{all}}$ in general. For example, $\neg\neg(p \vee q) \to (p \vee q)$ is not valid in $\mathsf{M}_{\mathrm{all}}$ (see [8, p.11]). Thus, together with our soundness result shown later, we can say that the inquisitive logic, i.e., all the formulas valid in $\mathsf{M}_{\mathrm{all}}$, is *not* closed under uniform substitutions.

The formula (ii) is usually called the Kreisel-Putnam Axiom. Since $?B := B \vee \neg B$, (ii) implies (iii) as a special case. The validity of (iii) has been called Mascarenhas Theorem [8, Theorem 3, see also footnote 16 in p.12]. (iv) says that every frame is of depth ≤ 2 over intuitionistic Kripke frames [4, Proposition 2.38.]. Finally, (v) says that every rooted subframe of the original frame is of width ≤ 2 over intuitionistic Kripke frames [4, Proposition 2.39.].

Proof. Here we show (i) alone. The crucial part is the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{0 : p \Rightarrow 0 : p}{1 : p, 2 : p \Rightarrow 0 : p}\,(\text{Atom L})}{2 : p \Rightarrow 0 : p, 1 : \neg p}\,(\neg\text{R}_{1,2})}{\Rightarrow 0 : p, 1 : \neg p, 2 : \neg p}\,(\neg\text{R}_{1,2})}{2 : \neg\neg p \Rightarrow 0 : p, 1 : \neg p}\,(\neg\text{L})}{1 : \neg\neg p, 2 : \neg\neg p \Rightarrow 0 : p}\,(\neg\text{L})}{0 : \neg\neg p \Rightarrow 0 : p}\,(\text{Move})$$

The rest of the proof can be found in Appendix A. □

4 Completeness

In this section, we show that the tree-sequent calculus $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$ is sufficient to prove all formulas that are valid in $\mathsf{M}_{\mathrm{all}}$.

In the following, Γ, Δ are possibly infinite in the expression $\Gamma \Rightarrow \Delta$ of a tree-sequent. In the case where Γ, Δ are all finite, the tree-sequent $\Gamma \Rightarrow \Delta$ said to be *finite*. A (possibly infinite) tree-sequent $\Gamma \Rightarrow \Delta$ is *provable* in $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$, if $\mathbf{cutfree}\mathbf{T}\mathsf{InqL} \vdash \Gamma' \Rightarrow \Delta'$ for some finite tree-sequent $\Gamma' \Rightarrow \Delta'$ such that $\Gamma' \subset \Gamma$ and $\Delta' \subset \Delta$. In what follows, we extend our notation $\mathbf{cutfree}\mathbf{T}\mathsf{InqL} \vdash \Gamma \Rightarrow \Delta$ to cover any possibly infinite tree-sequent in the sense explained above.

Definition 3 (Saturatedness). *A tree-sequent $\Gamma \Rightarrow \Delta$ is saturated, if it satisfies the following conditions:*

(consistency) (i) *If* $\alpha : A \in \Gamma$, *then* $\alpha : A \notin \Delta$, (ii) $\alpha : \bot \notin \Gamma$.
(hereditary condition) *If* $0 : A \in \Gamma$, *then* $1 : A \in \Gamma$ *and* $2 : A \in \Gamma$.
(atom l) *If* $1 : p \in \Gamma$ *and* $2 : p \in \Gamma$, *then* $0 : p \in \Gamma$.
($\wedge$l) *If* $\alpha : A \wedge B \in \Gamma$, *then* $\alpha : A \in \Gamma$ *and* $\alpha : B \in \Gamma$.
($\wedge$r) *If* $\alpha : A \wedge B \in \Delta$, *then* $\alpha : A \in \Delta$ *or* $\alpha : B \in \Delta$.
($\vee$l) *If* $\alpha : A \vee B \in \Gamma$, *then* $\alpha : A \in \Gamma$ *or* $\alpha : B \in \Gamma$.
($\vee$r) *If* $\alpha : A \vee B \in \Delta$, *then* $\alpha : A \in \Delta$ *and* $\alpha : B \in \Delta$.
($\neg$l) *If* $\alpha : \neg A \in \Gamma$, *then* $\alpha : A \in \Delta$.
($\neg\mathbf{r}_{1,2}$) *If* $\alpha : \neg A \in \Delta$ *and* $\alpha \neq 0$, *then* $\alpha : A \in \Gamma$.
($\neg\mathbf{r}_0$) *If* $0 : \neg A \in \Delta$, *then* $1 : A \in \Gamma$ *or* $2 : A \in \Gamma$.
($\rightarrow$l) *If* $\alpha : A \rightarrow B \in \Gamma$, *then* $\alpha : A \in \Delta$ *or* $\alpha : B \in \Gamma$.
($\rightarrow\mathbf{r}_{1,2}$) *If* $\alpha : A \rightarrow B \in \Delta$ *and* $\alpha \neq 0$, *then* $\alpha : A \in \Gamma$ *and* $\alpha : B \in \Delta$.
($\rightarrow\mathbf{r}_0$) *If* $0 : A \rightarrow B \in \Delta$, *then* $(0 : A \in \Gamma$ *and* $0 : B \in \Delta)$ *or* $(1 : A \in \Gamma$ *and* $1 : B \in \Delta)$ *or* $(2 : A \in \Gamma$ *and* $2 : B \in \Delta)$.

Lemma 1. *If a finite tree-sequent* $\Gamma \Rightarrow \Delta$ *is not provable in* **cutfreeT**lnqL, *then there exists a saturated tree-sequent* $\Gamma^+ \Rightarrow \Delta^+$ *such that* $\Gamma \subset \Gamma^+$ *and* $\Delta \subset \Delta^+$ *and* $\Gamma^+ \Rightarrow \Delta^+$ *is not provable in* **cutfreeT**lnqL.

The proof of this Lemma can be found in Appendix B. Recall that M_2 is the class of all inquisitive models whose domain consists of exactly two elements.

Theorem 1. *If* A *is valid in* M_2, *then* $\Rightarrow 0 : A$ *is provable in* **cutfreeT**lnqL. *Therefore, if* A *is valid in* $\mathsf{M}_{\mathrm{all}}$, *then* $\Rightarrow 0 : A$ *is provable in* **cutfreeT**lnqL.

Proof. It suffices to establish the first part. We show the contrapositive implication of it. Assume that $\Rightarrow 0 : A$ is unprovable in **cutfreeT**lnqL. Then, by Lemma 1, there exists some saturated tree-sequent $\Gamma^+ \Rightarrow \Delta^+$ such that $0 : A \in \Delta^+$ and **cutfreeT**lnqL $\nvdash \Gamma^+ \Rightarrow \Delta^+$. Let us define $W := \{1, 2\}$, $V(p) := \{\alpha \in \{1, 2\} \mid \alpha : p \in \Gamma^+\}$ $(p \in \wp)$. Now we show by induction on X of $\mathcal{L}_\wp$ that:

(i) If $0 : X \in \Gamma^+$, then $(1, 2) \models X$.
(ii) If $0 : X \in \Delta^+$, then $(1, 2) \not\models X$.
(iii) If $\alpha : X \in \Gamma^+$ and $\alpha \neq 0$, then $(\alpha, \alpha) \models X$.
(iv) If $\alpha : X \in \Delta^+$ and $\alpha \neq 0$, then $(\alpha, \alpha) \not\models X$.

Here we consider only the cases where X is of the form $p \in \wp$, of the form $\neg B$, and, of the form $B \rightarrow C$.

(X is of the form $p \in \wp$). We only show the cases (i) and (ii). (i) Suppose that $0 : p \in \Gamma^+$. Since $\Gamma^+ \Rightarrow \Delta^+$ is saturated, $1 : p$, $2 : p \in \Gamma^+$ by (**hereditary condition**). So, $1, 2 \in V(p)$. Thus, $(1, 2) \models p$. (ii) Suppose that $0 : p \in \Delta^+$. Since **cutfreeT**lnqL $\nvdash \Gamma^+ \Rightarrow \Delta^+$ and $\Gamma^+ \Rightarrow \Delta^+$ is saturated, $0 : p \notin \Gamma^+$ by (**consistency**). $0 : p \notin \Gamma^+$ means that $1 : p \notin \Gamma^+$ or $2 : p \notin \Gamma^+$ by (**atom l**). So, $1 \notin V(p)$ or $2 \notin V(p)$. Thus, we have $(1, 2) \not\models p$.

(X is of the form $\neg B$). We show the cases (i) and (ii) alone. (i) Suppose that $0 : \neg B \in \Gamma^+$. Since $\Gamma^+ \Rightarrow \Delta^+$ is saturated, $1 : \neg B \in \Gamma^+$ and $2 : \neg B \in \Gamma^+$ by (**hereditary condition**). By ($\neg$**l**), $1 : B, 2 : B \in \Delta^+$. By I.H., we have: $(1, 1) \not\models B$ and $(2, 2) \not\models B$.

It follows from Proposition 1 (iii) that $(1,2) \models \neg B$. (ii) Suppose that $0 : \neg B \in \Delta^+$. Since $\Gamma^+ \Rightarrow \Delta^+$ is saturated, $1 : B \in \Gamma^+$ or $2 : B \in \Gamma^+$ by ($\neg \mathbf{r}_0$). We deduce from I.H. that $(1,1) \models B$ or $(2,2) \models B$. By Proposition 1 (iii), $(1,2) \models \neg A$.

(X is of the form $B \to C$). We only show the cases (i) and (ii). (i) Suppose that $0 : B \to C \in \Gamma^+$. Since $\Gamma^+ \Rightarrow \Delta^+$ is saturated, $0 : B \in \Delta^+$ or $0 : C \in \Gamma^+$ by ($\to$**l**). We deduce from I.H. that $(1,2) \models B$ implies $(1,2) \models C$. Next, we obtain $1 : B \to C$ and $2 : B \to C \in \Gamma^+$ by (**hereditary condition**) and our assumption. Also by I.H. and ($\to$**l**), we get: $(1,1) \models B$ implies $(1,1) \models C$, and, $(2,2) \models B$ implies $(2,2) \models C$. Therefore, we conclude that $(1,2) \models B \to C$. (ii) Suppose that $0 : B \to C \in \Delta^+$. By ($\to \mathbf{r}_0$), one of the following holds: (a) $0 : A \in \Gamma^+$ and $0 : B \in \Delta^+$; (b) $1 : A \in \Gamma^+$ and $1 : B \in \Delta^+$; (c) $2 : A \in \Gamma^+$ and $2 : B \in \Delta^+$. By I.H., we establish that $(1,2) \not\models B \to C$.

Since $0 : A \in \Delta^+$, we have $(1,2) \not\models A$. Therefore, A is not valid in this finite model $\langle W, V \rangle \in \mathsf{M}_2$. □

5 Soundness

In this section, we establish that the tree-sequent calculus $\mathbf{T}\mathsf{InqL}$ (i.e., $\mathbf{cutfree}\mathbf{T}\mathsf{InqL}$ with (Cut)) is sound with respect to the class $\mathsf{M}_{\mathrm{all}}$ of all inquisitive models.

Each node α of a tree-sequent $\Gamma \Rightarrow \Delta$ is associated with a sequent $\Gamma_\alpha \Rightarrow \Delta_\alpha$ where Γ_α (or, Δ_α) is the set of formulas such that $\alpha : A \in \Gamma$ (or, $\alpha : A \in \Delta$, respectively). We define a translation of tree-sequents into formulas of $\mathcal{L}_\wp$. In the following, tree-sequents are all finite.

Definition 4 (Formulaic Translation). *Let $\Gamma \Rightarrow \Delta$ be a tree-sequent and s, t be fresh propositional variables in $\Gamma \Rightarrow \Delta$. The formulaic translation $[\![\Gamma \Rightarrow \Delta]\!]$ is defined as:*

$$[\![\Gamma \Rightarrow \Delta]\!] \equiv \bigwedge \Gamma_0 \to \Big((s \vee t) \vee \bigvee \Delta_0 \vee [\![\Gamma \Rightarrow \Delta]\!]_1 \vee [\![\Gamma \Rightarrow \Delta]\!]_2 \Big) \text{ where:}$$

$$[\![\Gamma \Rightarrow \Delta]\!]_1 \equiv s \wedge \bigwedge \Gamma_1 \to t \vee \bigvee \Delta_1; \quad [\![\Gamma \Rightarrow \Delta]\!]_2 \equiv t \wedge \bigwedge \Gamma_2 \to s \vee \bigvee \Delta_2.$$

Note that this formulaic translation depends on the choice of s and t.

Remark 2. Recall from Definition 2 that we use the fixed tree $\mathcal{T}$ unlike the previous studies [9,10,11] which employ 'growing' tree-sequents. This distinction makes us to use some fresh propositional variables in our formulaic translation. This is because the naive formulaic translation with no fresh variables:

$$[\![\Gamma \Rightarrow \Delta]\!]' \equiv \bigwedge \Gamma_0 \to \Big(\bigvee \Delta_0 \vee [\![\Gamma \Rightarrow \Delta]\!]_1 \vee [\![\Gamma \Rightarrow \Delta]\!]_2 \Big) \text{ where:}$$

$$[\![\Gamma \Rightarrow \Delta]\!]'_1 \equiv \bigwedge \Gamma_1 \to \bigvee \Delta_1; \quad [\![\Gamma \Rightarrow \Delta]\!]'_2 \equiv \bigwedge \Gamma_2 \to \bigvee \Delta_2.$$

does not preserve validity, e.g., when we apply (Atom L) (cf. Lemma 3 (**atom left**)).

One more difference. The previous studies mentioned above used the formulaic translation to connect the tree-sequent calculus with the intended Hilbert calculus and showed that the translation preserves the provability. We, however, use it to establish the soundness.

From the definition of $[\![\Gamma \Rightarrow \Delta]\!]$, we can easily derive the following.

Lemma 2. *If* $[\![\Gamma \Rightarrow \Delta]\!]_\alpha$ *is valid in* $\mathsf{M}_{\mathrm{all}}$ *for some* $\alpha \in \{1, 2\}$*, then* $[\![\Gamma \Rightarrow \Delta]\!]$ *is valid in* $\mathsf{M}_{\mathrm{all}}$.

Lemma 3. *The following formulas are valid in* $\mathsf{M}_{\mathrm{all}}$.

(ax) $A \wedge C \to A \vee D$.
($\bot$ left) $\bot \wedge C \to D$.
(atom left) $X_1 \to X_2$, *where:*

$$X_1 \equiv p \to (S \vee T) \vee D \vee (S \wedge E \to T \vee F) \vee (T \wedge G \to S \vee H);$$
$$X_2 \equiv (S \vee T) \vee D \vee (p \wedge S \wedge E \to T \vee F) \vee (p \wedge T \wedge G \to S \vee H).$$

(move) $((E \wedge A \to F) \vee (G \wedge A \to H)) \to (A \to (E \to F) \vee (G \to H))$.
($\wedge$ right) $(C \to D \vee A) \wedge (C \to D \vee B) \to (C \to (D \vee (A \wedge B)))$.
($\vee$ left) $(A \wedge C \to D) \wedge (B \wedge C \to D) \to (((A \vee B) \wedge C) \to D)$.
($\neg$ left) $(C \to D \vee A) \to (\neg A \wedge C \to D)$.
($\neg$ right$_{1,2}$) $(C \wedge A \to D) \to (C \to D \vee \neg A)$.
($\neg$ right$_0$) $X_3 \wedge X_4 \to X_5$, *where:*

$$X_3 \equiv (S \vee T) \vee D \vee (S \wedge E \wedge A \to F \vee T) \vee (T \wedge G \to S \vee H);$$
$$X_4 \equiv (S \vee T) \vee D \vee (S \wedge E \to F \vee T) \vee (T \wedge G \wedge A \to S \vee H);$$
$$X_5 \equiv (S \vee T) \vee \neg A \vee D \vee (S \wedge E \to F \vee T) \vee (T \wedge G \to S \vee H).$$

($\to$ left) $(C \to D \vee A) \wedge (C \wedge B \to D) \to (C \wedge (A \to B) \to D)$.
($\to$ right$_{1,2}$) $(C \wedge A \to D \vee B) \to (C \to (D \vee (A \to B)))$.
($\to$ right$_0$) $(X_6 \wedge X_7 \wedge X_8) \to X_9$, *where:*

$$X_6 \equiv A \to ((S \vee T) \vee D \vee B \vee (S \wedge E \to T \vee F) \vee (T \wedge G \to S \vee H));$$
$$X_7 \equiv (S \vee T) \vee D \vee (S \wedge E \wedge A \to T \vee F) \vee (T \wedge G \to S \vee H);$$
$$X_8 \equiv (S \vee T) \vee D \vee (S \wedge E \to T \vee F) \vee (T \wedge G \wedge A \to S \vee H);$$
$$X_9 \equiv (S \vee T) \vee (A \to B) \vee D \vee (S \wedge E \to T \vee F) \vee (T \wedge G \to S \vee H).$$

(cut) $(C \to D \vee A) \wedge (C \wedge A \to D) \to (C \to D)$.

Proof. Formulas except (**atom left**), ($\neg$ **right**$_0$) and ($\to$ **right**$_0$) are all theorems of intuitionistic logic. Therefore, they are all valid in $\mathsf{M}_{\mathrm{all}}$ (recall the description just after Proposition 5). So, it suffices to check (**atom left**), ($\neg$ **right**$_0$) and ($\to$ **right**$_0$). For more details, see Appendix C. □

Lemma 4. *If* $\Gamma \Rightarrow \Delta$ *is provable in* $\mathbf{T}\mathsf{InqL}$*, then* $[\![\Gamma \Rightarrow \Delta]\!]$ *is valid in* $\mathsf{M}_{\mathrm{all}}$.

Proof. By induction on the derivation of $\Gamma \Rightarrow \Delta$ in $\mathbf{T}\mathsf{InqL}$. First of all, let us choose some fresh propositional variables s, t not occurring in the derivation. We assume that all formulaic translations in this proof depend on s and t. All cases immediately follow from Lemmas 2 and 3. □

In order to establish the soundness through our formulaic translation with fresh variables, we need to show the following, which lets us use the fresh propositional variables s and t to *name* three worlds (corresponding to 0, 1, 2 in our fixed tree) in an inquisitive model [1].

Lemma 5. *Assume that s and t do not occur in A. If $(s \vee t) \vee A \vee (s \to t) \vee (t \to s)$ is valid in $\mathsf{M}_{\mathrm{all}}$, then A is valid in $\mathsf{M}_{\mathrm{all}}$.*

Proof. We prove the contrapositive implication. Assume that A is not valid in $\mathsf{M}_{\mathrm{all}}$. By Proposition 5, there exists some inquisitive model $\mathfrak{M} = \langle W, V \rangle$ and $w, v \in W$ such that $w \neq v$ and $\#W \geq 2$ and $(w, v) \not\models_{\mathfrak{M}} A$. Let us define a valuation V' extending V by: $V'(p) := \{ w \}$ (if $p \equiv s$); $\{ v \}$ (if $p \equiv t$); $V(p)$ (o.w.). Write $\mathfrak{M}' = \langle W, V' \rangle$. Then, $(x, y) \models_{\mathfrak{M}} B$ iff $(x, y) \models_{\mathfrak{M}'} B$, for any $x, y \in W$ and any subformula B of A. Thus, $(w, v) \not\models_{\mathfrak{M}'} A$. By definition of V', $(w, v) \not\models_{\mathfrak{M}'} (s \vee t) \vee (s \to t) \vee (t \to s)$. Therefore, $(w, v) \not\models_{\mathfrak{M}'} (s \vee t) \vee A \vee (s \to t) \vee (t \to s)$. □

Theorem 2. *If $\Rightarrow 0 : A$ is provable in $\mathbf{T}\mathsf{InqL}$, A is valid in $\mathsf{M}_{\mathrm{all}}$.*

Proof. By Lemma 4, $[\![\Rightarrow 0 : A]\!]$ is valid in $\mathsf{M}_{\mathrm{all}}$, i.e., $(s \vee t) \vee A \vee (s \to t) \vee (t \to s)$ is valid in $\mathsf{M}_{\mathrm{all}}$. It follows from Lemma 5 that A is valid in $\mathsf{M}_{\mathrm{all}}$. □

6 Cut-Elimination and Decidability of $\mathbf{T}\mathsf{InqL}$

Corollary 1. *The following are equivalent:* (i) $\mathbf{cutfree}\mathbf{T}\mathsf{InqL} \vdash \Rightarrow 0 : A$; (ii) $\mathbf{T}\mathsf{InqL} \vdash \Rightarrow 0 : A$; (iii) *$A$ is valid in $\mathsf{M}_{\mathrm{all}}$;* (iv) *$A$ is valid in M_2.*

Proof. [(i) ⇒ (ii)] Trivial. [(ii) ⇒ (iii)] This follows from Theorem 2. [(iii) ⇒ (iv)] Easy. [(iv) ⇒ (i)] This follows from Theorem 1. □

By this corollary, we semantically establish that $\mathbf{T}\mathsf{InqL}$ enjoys cut-elimination, i.e., (ii) ⇒ (i). Furthermore, this corollary tells us that $\mathbf{T}\mathsf{InqL}$ is decidable.

References

1. Areces, C., ten Cate, B.: Hybrid logics. In: Blackburn, P., van Benthem, J., Wolter, F. (eds.) Handbook of Modal Logic, pp. 821–868. Elsevier, Amsterdam (2007)
2. Blackburn, P., ten Cate, B.: Pure extensions, proof rules, and hybrid axiomatics. Studia Logica 84, 277–322 (2006)
3. Ten Cate, B., Shan, C.: Axiomatizing Groenendijk's logic of interrogation. In: Aloni, M., Butler, A., Dekker, P. (eds.) Questions in Dynamic Semantics, pp. 63–82. Elsevier, Oxford (2007)
4. Chagrov, A., Zakharyaschev, M.: Modal Logic. Oxford Logic Guides, vol. 35. Oxford Science Publications, Oxford (1997)
5. Ciardelli, I., Roelofsen, F.: Generalized inquisitive logic (submitted for publication, 2009)
6. Gabbay, D.: Labelled Deductive Systems. Clarendon Press, Oxford (1996)

[1] Some readers who are familiar with *hybrid logic* [1] may find the similarity with the proof-rule called $\mathbf{Name}'$ [2, p.294].

7. Groenendijk, J.: The logic of interrogation. In: Matthews, T., Strolovitch, D. (eds.) The Proceedings of the Ninth Conference on Semantics and Linguistic Theory, pp. 109–126. CLC Publications, NY (1999)
8. Groenendijk, J.: Inquisitive semantics: Two possibilities for disjunction. Prepublication Series PP-2008-26, ILLC (2008)
9. Hasuo, I., Kashima, R.: Kripke completeness of first-order constructive logics with strong negation. Logic Journal of the IGPL 11(6), 615–646 (2003)
10. Ishigaki, R., Kikuchi, K.: A tree-sequent calculus for a natural predicate extension of Visser's propositional logic. Logic Journal of the IGPL 15(2), 149–164 (2007)
11. Ishigaki, R., Kashima, R.: Sequent calculi for some strict implication logics. Logic Journal of the IGPL 16(2), 155–174 (2008)
12. Kashima, R.: Sequent calculi of non-classical logics - Proofs of completeness theorems by sequent calculi (in Japanese). In: Proceedings of Mathematical Society of Japan Annual Colloquium of Foundations of Mathematics, pp. 49–67 (1999)
13. Mascarenhas, S.: Inquisitive semantics and logic. The Institute for Logic, Language and Computation, the University of Amsterdam, Forthcoming Master Thesis
14. Mascarenhas, S.: Inquisitive semantics and logic. ILLC, Amsterdam (2008) (manuscript)

A Some Examples of Derivations in $\mathbf{TInqL}$

Let us finish proving Proposition 6. The remaining items are:

(ii) $(\neg A \to B \vee C) \to (\neg A \to B) \vee (\neg A \to C)$
(iii) $(\neg A \to ?B) \to (\neg A \to B) \vee (\neg A \to \neg B)$
(iv) $A \vee (A \to (B \vee \neg B))$
(v) $(A \to (B \vee C)) \vee (B \to (A \vee C)) \vee (C \to (A \vee B))$

It suffices to show (ii), (iv) and (v) are provable in $\mathbf{cutfreeTInqL}$. As for (ii), the crucial part in the following derivation:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \vdots\\ \Theta \Rightarrow 1:B, 2:C, 0:\neg A\end{array} \qquad \dfrac{\dfrac{\Theta, 1:B, 2:B \Rightarrow 1:B, 2:C}{\Theta, 0:B \Rightarrow 1:B, 2:C}\,(\text{Move}) \quad \dfrac{\Theta, 1:B, 1:C \Rightarrow 1:B, 2:C}{\Theta, 0:C \Rightarrow 1:B, 2:C}\,(\text{Move})}{\Theta, 0:B \vee C \Rightarrow 1:B, 2:C}\,(\vee\text{L})}{\Theta, 0:\neg A \to B \vee C \Rightarrow 1:B, 2:C}\,(\to\text{L})$$

where $\Theta := \{\, 1:\neg A, 2:\neg A \,\}$ and $\mathcal{D}$ is:

$$\dfrac{\dfrac{1:A, 2:\neg A \Rightarrow 1:B, 2:C, 1:A}{1:A, \Theta \Rightarrow 1:B, 2:C}\,(\neg\text{L}) \quad \dfrac{2:A, 1:\neg A \Rightarrow 1:B, 2:C, 2:A}{2:A, \Theta \Rightarrow 1:B, 2:C}\,(\neg\text{L})}{\Theta \Rightarrow 1:B, 2:C, 0:\neg A}\,(\neg\text{R}_0)$$

We can establish (iv) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{1:B, 1:A \Rightarrow 0:A, 1:B}{1:A \Rightarrow 0:A, 1:B, 1:\neg B}\,(\neg\text{R}_{1,2})}{1:A \Rightarrow 0:A, 1:B \vee \neg B}\,(\vee\text{R}) \quad \dfrac{\dfrac{2:B, 2:A \Rightarrow 0:A, 2:B}{2:A \Rightarrow 0:A, 2:B, 2:\neg B}\,(\neg\text{R}_{1,2})}{2:A \Rightarrow 0:A, 2:B \vee \neg B}\,(\vee\text{R}) \qquad 0:A \Rightarrow 0:A, 0:B \vee \neg B}{\Rightarrow 0:A, 0:A \to (B \vee \neg B)}\,(\to\text{R}_0)}{\Rightarrow 0:A \vee (A \to (B \vee \neg B))}\,(\vee\text{R})$$

As for (v), the crucial part is the following derivation:

$$\dfrac{\dfrac{1:A, 2:B, 1:B, 2:C \Rightarrow 0:B \vee C, 1:A, 1:C, 2:A \vee B}{0:A, 1:B, 2:C \Rightarrow 0:B \vee C, 1:A, 1:C, 2:A \vee B}\,(\text{Move})}{0:A, 1:B, 2:C \Rightarrow 0:B \vee C, 1:A \vee C, 2:A \vee B}\,(\vee\text{R})$$

B A Proof of Lemma 1

Here, we would like to give a proof of Lemma 1.

Proof. Suppose that a finite tree-sequent $\Gamma \Rightarrow \Delta$ is not provable in $\mathbf{cutfreeTlnqL}$. In the following, we construct an infinite sequence of finite tree-sequents $(\Gamma^i \Rightarrow \Delta^i)_{i\in\omega}$ and obtain $\Gamma^+ \Rightarrow \Delta^+$ as the union of them.

Let $(\alpha_i : F_i)_{i>1}$ be an enumeration of all labelled formulas such that each formula of $\mathcal{L}_\wp$ appears infinitely many times. From now on, we construct $(\Gamma^i \Rightarrow \Delta^i)_{i\in\omega}$ such that $\mathbf{cutfreeTlnqL} \nvdash \Gamma^i \Rightarrow \Delta^i$.
(Basis) Let $\Gamma^0 \Rightarrow \Delta^0 \equiv \Gamma \Rightarrow \Delta$. By assumption, $\mathbf{cutfreeTlnqL} \nvdash \Gamma^0 \Rightarrow \Delta^0$.
(Inductive step) Suppose that we have already defined $\Gamma^{k-1} \Rightarrow \Delta^{k-1}$ such that $\mathbf{cutfreeTlnqL} \nvdash \Gamma^{k-1} \Rightarrow \Delta^{k-1}$. In this k-th step, we define $\Gamma^k \Rightarrow \Delta^k$ so that unprovability of the tree-sequent is preserved. The operations executed in the k-th step are as follows:

- First, for any $0 : A \in \Gamma^k$, we add $1 : A$ and $2 : A$ to Γ^{k-1}. Unprovability is preserved because of the rule (Move). We denote the result of this step by $(\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$.
- Second, according to the form of $\alpha_k : F_k$, one of the following operation is executed:
  (1) The case where $F_k \equiv p$ and $\alpha_k \neq 0$ and $\alpha_k : F_k \in (\Gamma^{k-1})'$. Define:

$$\Gamma^k \Rightarrow \Delta^k \equiv \begin{cases} 0 : p, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1} & \text{if } (3 - \alpha_k) : p \in (\Gamma^{k-1})'; \\ (\Gamma^{k-1})' \Rightarrow \Delta^{k-1} & \text{o.w.} \end{cases}$$

  Unprovability is preserved because of (Atom L).
  (2) The case where $F_k \equiv A \wedge B$ and $\alpha_k : F_k \in (\Gamma^{k-1})'$. Define $\Gamma^k \Rightarrow \Delta^k \equiv \alpha_k : A, \alpha_k : B, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$. Unprovability is preserved because of ($\wedge$L).
  (3) The case where $F_k \equiv A \wedge B$ and $\alpha_k : F_k \in \Delta^{k-1}$. By ($\wedge$R), we know that either $(\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, \alpha_k : A$ or $(\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, \alpha_k : B$ is unprovable. So, choose an unprovable tree-sequent as $\Gamma^k \Rightarrow \Delta^k$.
  (4) The case where $F_k \equiv A \vee B$ and $\alpha_k : F_k \in (\Gamma^{k-1})'$. Similar to (3).
  (5) The case where $F_k \equiv A \vee B$ and $\alpha_k : F_k \in \Delta^{k-1}$. Similar to (2).
  (6) The case where $F_k \equiv \neg A$ and $\alpha_k : F_k \in (\Gamma^{k-1})'$. Define $\Gamma^k \Rightarrow \Delta^k \equiv (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, \alpha_k : A$. Unprovability is preserved because of ($\neg$L).
  (7) The case where $F_k \equiv \neg A$ and $\alpha_k : F_k \in \Delta^{k-1}$.
    - (α_k = 1 or 2) Define $\Gamma^k \Rightarrow \Delta^k \equiv \alpha_k : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$. Unprovability is preserved because of ($\neg \mathrm{R}_{1,2}$).
    - (α_k = 0) By ($\neg \mathrm{R}_0$), we know that either $1 : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$ or $2 : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$ is unprovable. So, choose an unprovable tree-sequent as $\Gamma^k \Rightarrow \Delta^k$.

(8) The case where $F_k \equiv A \to B$ and $\alpha_k : F_k \in (\Gamma^{k-1})'$. By ($\to$L), we know that either $\alpha_k : B, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$ or $(\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, \alpha_k : A$ is unprovable. Thus, choose an unprovable tree-sequent as $\Gamma^k \Rightarrow \Delta^k$.

(9) The case where $F_k \equiv A \to B$ and $\alpha_k : F_k \in \Delta^{k-1}$.
- (α_k = 1 or 2) Define $\Gamma^k \Rightarrow \Delta^k \equiv \alpha_k : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, \alpha_k : B$. Unprovability is preserved because of ($\to \mathrm{R}_{1,2}$).
- (α_k = 0) By ($\neg \mathrm{R}_0$), we know that none of $0 : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, 0 : B$ or $1 : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, 1 : B$ or $2 : A, (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}, 2 : B$ are provable. So, choose an unprovable tree-sequent as $\Gamma^k \Rightarrow \Delta^k$.

(10) Otherwise. It suffices to define $\Gamma^k \Rightarrow \Delta^k \equiv (\Gamma^{k-1})' \Rightarrow \Delta^{k-1}$.

Now let $\Gamma^+ \Rightarrow \Delta^+$ be $(\bigcup_{i\in\omega} \Gamma^i) \Rightarrow (\bigcup_{i\in\omega} \Delta^i)$. It is easy to verify that the tree-sequent $\Gamma^+ \Rightarrow \Delta^+$ is saturated. □

C A Proof of Lemma 3

Here we would like to give a detailed proof of Lemma 3.

Proof. Let us check (**atom left**), ($\neg$ **right**$_0$) and ($\to$ **right**$_0$) one by one. By Proposition 5, consider any $\langle W, V \rangle$ with $\#W \geq 2$ and $v, w \in W$ with $w \neq v$.

(atom left) We show $(w, v) \models X_1 \to X_2$. Assume for contradiction that $(w, v) \not\models X_1 \to X_2$. Remark that $X_1 \to X_2$ is a truth-functional tautology (it suffices to note that $P \to Q$ is equivalent to $\neg P \vee Q$ in classical logic). Thus, $(w, v) \models X_1$ and $(w, v) \not\models X_2$ by Proposition 2. From $(w, v) \not\models X_2$, we obtain (a) $(w, v) \not\models (S \vee T) \vee D$, (b) $(w, v) \not\models (p \wedge S \wedge E) \to (T \vee F)$ and (c) $(w, v) \not\models (p \wedge T \wedge G) \to (S \vee H)$. In view of (a), we can restrict our attention to the following essential case where we have (b′) $(w, w) \models p \wedge S \wedge E$ and $(w, w) \not\models T \vee F$ and (c′) $(v, v) \models p \wedge T \wedge G$ and $(v, v) \not\models S \vee H$. Since $(w, w) \models p$ and $(v, v) \models p$, we have $(w, v) \models p$. It follows from $(w, v) \models X_1$ and (a) that $(w, v) \models (S \wedge E \to T \vee F) \vee (T \wedge G \to S \vee H)$. Thus, we subdivide our argument into the following two cases: (Case 1) Assume that $(w, v) \models S \wedge E \to T \vee F$. By Proposition 1 (ii), we have $(w, w) \models S \wedge E \to T \vee F$. It follows from (b′) that $(w, w) \not\models S \vee E \to T \vee F$. Contradiction. (Case 2) Assume that $(w, v) \models T \wedge G \to S \vee H$. Similarly to (Case 1), we get the desired contradiction.

($\neg$ right$_0$) We show that $(w, v) \models X_3 \wedge X_4 \to X_5$. Assume for contradiction that $(w, v) \not\models X_3 \wedge X_4 \to X_5$. Similarly to the proof of (**atom left**), remark that $X_3 \wedge X_4 \to X_5$ is a truth-functional tautology. Thus, $(w, v) \models X_3 \wedge X_4$ and $(w, v) \not\models X_5$ by Proposition 2. From $(w, v) \not\models X_5$, we have (a) $(w, v) \not\models (S \vee T) \vee D$, (b) $(w, v) \not\models \neg A$, (c) $(w, v) \not\models (S \wedge E) \to (T \vee F)$, (d) $(w, v) \not\models (T \wedge G) \to (S \vee H)$. We can derive from (a) that (c′) $(w, w) \models S \wedge E$ and $(w, w) \not\models T \vee F$ and (d′) $(v, v) \models T \wedge G$ and $(v, v) \not\models S \vee H$. By $(w, v) \models X_3 \wedge X_4$, (a), (c), and (d), we derive (e) $(w, v) \models S \wedge E \wedge A \to F \vee T$ and (f) $(w, v) \models T \wedge G \wedge A \to S \vee H$. From (b) and Proposition 1 (iii), we have (Case 1) $(w, w) \models A$ or (Case 2) $(v, v) \models A$. Since we can prove (Case 2) similarly to (Case 1), it suffice to check (Case 1). Assume that $(w, w) \models A$. By Proposition 1 and $(w, v) \models S \wedge E \wedge A \to F \vee T$, $(w, w) \models S \wedge E \to F \vee T$. Together with our assumption and (c′), we obtain $(w, w) \not\models S \wedge E \to F \vee T$. Contradiction.

($\rightarrow$ right$_0$) We show that $(w, v) \models X_6 \wedge X_7 \wedge X_8 \rightarrow X_9$. Assume for contradiction that $(w, v) \not\models X_6 \wedge X_7 \wedge X_8 \rightarrow X_9$. Since $X_6 \wedge X_7 \wedge X_8 \rightarrow X_9$ is a truth-functional tautology, we derive that $(w, v) \models X_6 \wedge X_7 \wedge X_8$ but $(w, v) \not\models X_9$ by Proposition 2. It follows from $(w, v) \not\models X_9$ that: (a) $(w, v) \not\models (S \vee T) \vee D$; (b) $(w, v) \not\models A \rightarrow B$; (c) $(w, v) \not\models (S \wedge E) \rightarrow (T \vee F)$; (d) $(w, v) \not\models (T \wedge G) \rightarrow (S \vee H)$. We can derive from (a) that (c′) $(w, w) \models S \wedge E$ and $(w, w) \not\models T \vee F$ and (d′) $(v, v) \models T \wedge G$ and $(v, v) \not\models S \vee H$. Since $(w, v) \models X_7$ and $(w, v) \models X_8$, we derive (e) $(w, v) \models S \wedge E \wedge A \rightarrow F \vee T$ and (f) $(w, v) \models T \wedge G \wedge A \rightarrow S \vee H$. By (b), we subdivide our argument into the following three cases: (Case 1) $(w, v) \models A$ but $(w, v) \not\models B$; (Case 2) $(w, w) \models A$ but $(w, w) \not\models B$; (Case 3) $(v, v) \models A$ but $(v, v) \not\models B$. Because we can prove (Case 3) similarly to (Case 2), we only check (Case 1) and (Case 2).

- (Case 1) Since $(w, v) \models X_6$, we get $(w, v) \models (S \vee T) \vee D \vee B \vee (S \wedge E \rightarrow T \vee E) \vee (T \wedge G \rightarrow S \vee H)$, which implies a contradiction.
- (Case 2) It follows from $(w, w) \models X_6$ and Proposition 1 (ii), and that $(w, w) \models S \wedge E \wedge A \rightarrow T \vee E$. Since $(w, w) \models A$, we have $(w, w) \models S \wedge E \rightarrow T \vee E$, which contradicts (c'). □

The Arrow Calculus as a Quantum Programming Language

Juliana Kaizer Vizzotto[1], André Rauber Du Bois[2], and Amr Sabry[3]

[1] Mestrado em Nanociências, Centro Universitário Franciscano
Santa Maria, RS/ Brazil
[2] PPGI, Universidade Católica de Pelotas
Pelotas, RS/Brazil
[3] Department of Computer Science, Indiana University
Bloomington, USA

Abstract. We express quantum computations (with measurements) using the arrow calculus extended with monadic constructions. This framework expresses quantum programming using well-understood and familiar classical patterns for programming in the presence of computational effects. In addition, the five laws of the arrow calculus provide a convenient framework for *equational reasoning* about quantum computations that include measurements.

1 Introduction

Quantum computation [1] can be understood as a *transformation* of information encoded in the state of a *quantum* physical system. Its basic idea is to encode data using quantum bits (qubits). Differently from the classical bit, a qubit can be in a *superposition* of basic states leading to "quantum parallelism." This form of parallelism is due to the non-local wave character of quantum information and is qualitatively different from the classical notion of parallelism. This characteristic of quantum computation can greatly increase the processing speed of algorithms. However, quantum data types are computationally very powerful not only due to superposition. There are other odd properties like *measurement*, in which the observed part of the quantum state and every other part that is *entangled* with it immediately lose their wave character.

These interesting properties have led to the development of very efficient quantum algorithms, like Shor's quantum algorithm for factorizing integers [2], and Grover's quantum search on databases [3]. Another important theme is the development of quantum cryptographic techniques [4].

Since these discoveries, much research has been done on quantum computation. Summarizing the field of research we can classify it according three main areas: i) physical implementations of quantum computers, ii) development of new quantum algorithms; and iii) design of quantum programming languages.

This work is about the design of a quantum programming language, and consequently about a high-level, structured and well-defined way to develop new quantum algorithms and to *reason* about them.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 379–393, 2009.

We have been working on semantic models for quantum programming. In previous work [5] we established that general quantum computations (including measurements) are an instance of the category-theoretic concept of arrows [6], a generalization of *monads* [7] and *idioms* [8]. Translating this insight to a practical programming paradigm has been difficult however. On one hand, directly using arrows is highly non-intuitive, requiring programming in the so-called "point-free" style where intermediate computations are manipulated without giving them names. Furthermore reasoning about arrow programs uses nine, somewhat idiosyncratic laws.

In recent work, Lindley *et. al.* [9] present the *arrow calculus*, which is a more friendly version of the original presentation of arrows. The arrow calculus augment the simply typed lambda calculus with four constructs satisfying five laws. Two of these constructs resemble function abstraction and application, and satisfy familiar beta and eta laws. The remaining two constructs resemble the unit and bind of a monad, and satisfy left unit, right unit, and associativity laws. Basically, using the arrow calculus we can understand arrows through classic well-known patterns.

In this work we propose to express quantum computations using the arrow calculus axtended with monadic constructions. We show that quantum programming can be expressed using well-understood and familiar classical patterns for programming in the presence of computational effects. Interestingly, the five laws of the arrow calculus provide a convenient framework for *equational reasoning* about quantum computations (including measurements).

This work is organized as follows. The next two sections review the background material on modeling quantum computation using classical arrows. Section 4 presents the *arrow calculus*. We show the quantum arrow calculus in Section 5. We express some traditional examples of quantum computations using the quantum calculus. Additionally, we illustrate how we can use the calculus to reason about quantum programs. Section 6 concludes with a discussion of some related works. Finally, Appendix A presents the constructs of simply-typed lambda calculus, Appendix B gives an extension of the simply-typed lambda calculus with monadic constructions, and Appendix C reviews general quantum computations.

2 Classic Arrows

The simply-typed lambda calculus is an appropriate model of pure functional programming (see Appendix A). The standard way to model programming in the presence of effects is to use *monads* [10] (see Appendix B). Arrows, like monads, are used to elegantly program notions of computations in a pure functional setting. But unlike the situation with monads, which wrap the *results of computations*, arrows wrap the *computations* themselves.

From a programming point of view, *classic arrows* extend the simply-typed lambda calculus with one type and three constants satisfying nine laws (see Figure 1). The type $A \rightsquigarrow B$ denotes a computation that accepts a value of type A and returns a value of type B, possibly performing some side effects. The

Types

$arr :: (A \to B) \to (A \rightsquigarrow B)$
$(\ggg) :: (A \rightsquigarrow B) \to (B \rightsquigarrow C) \to (A \rightsquigarrow C)$
$first :: (A \rightsquigarrow B) \to (A \times C \rightsquigarrow B \times C)$

Definitions

$second : (A \rightsquigarrow B) \to (C \times A \rightsquigarrow C \times B)$
$second = \lambda f.arr\ swap \ggg first\ f \ggg arr\ swap$
$(\&\&\&) : (C \rightsquigarrow A) \to (C \rightsquigarrow B) \to (C \rightsquigarrow A \times B)$
$(\&\&\&) = \lambda f.\lambda g.arr\ sup \ggg first\ f \ggg second\ g$

Equations

$(\rightsquigarrow_1)\ arr\ id \ggg f = f$
$(\rightsquigarrow_2)\ f \ggg arr\ id = f$
$(\rightsquigarrow_3)\ (f \ggg g) \ggg h = f \ggg (g \ggg h)$
$(\rightsquigarrow_4)\ arr(g.f) = arr\ f \ggg arr\ g$
$(\rightsquigarrow_5)\ first(arr\ f) = arr(f \times id)$
$(\rightsquigarrow_6)\ first(f \ggg g) = first\ f \ggg first\ g$
$(\rightsquigarrow_7)\ first\ f \ggg arr(id \times g) = arr(id \times g) \ggg first\ f$
$(\rightsquigarrow_8)\ first\ f \ggg arr\ fst = arr\ fst \ggg f$
$(\rightsquigarrow_9)\ first(first\ f) \ggg arr = arr\ assoc \ggg first\ f$

Fig. 1. Classic Arrows

three constants are: *arr*, which promotes a function to a pure arrow with no side effects; **¿**, which composes two arrows; and *first*, which extends an arrow to act on the first component of a pair leaving the second component unchanged.

To understand the nine equations, we use some auxiliary functions. The function *second*, is like *first*, but acts on the second component of a pair, and $f \&\&\& g$, applies arrow f and g to the same argument and then pairs the results.

3 Quantum Arrows

Quantum computation is generally expressed in the framework of a Hilbert space (see Appendix C for a short review of that model). As expressive and as convenient is this framework for mathematical reasoning, it is not easily amenable to familiar programming techniques and abstractions. In recent work [5] however, we established that this general model of quantum computations (including measurements) can be structured using the category-theoretic concept of arrows. Figure 2 explains the main ideas which we elaborate on in the remainder of this section.

In the figure, we have added type definitions (i.e, type synonyms) for convenience. Type **Vec** A means that a vector is a function mapping elements from a vector space orthonormal basis to complex numbers (i.e., to their probability amplitudes). Type **Lin** represents a linear operator (e.g, a unitary matrix) mapping a vector of type A to a vector of type B. Note that if we *uncurry* the arguments A and B, it turns exactly into a square matrix (i.e, **Vec** (A, B)). Type **Dens** A stands for density matrices and it is straight to build from **Vec**. Type **Super** A B means a superoperator mapping a density matrix of type A

Type Definitions
$type\ \mathbf{Vec}\ A \quad = A \to \mathbb{C}$
$type\ \mathbf{Lin}\ A\ B \quad = A \to \mathbf{Vec}\ B$
$type\ \mathbf{Dens}\ A \quad = \mathbf{Vec}\ (A, A)$
$type\ \mathbf{Super}\ A\ B = (A, A) \to \mathbf{Dens}\ B$
Syntax
Types $A, B, C\ ::= ...\ \mathbf{Vec}\ A \mid \mathbf{Lin}\ A \mid \mathbf{Dens}\ A \mid \mathbf{Super}\ A\ B$
Terms $L, M, N ::= ... \mid return \mid \gg\!= \mid arr \mid \ggg \mid first$
Monadic Definitions
$return : A \to \mathbf{Vec}\ A$
$return\ a\ b = \mathsf{if}\ a == b\ \mathsf{then}\ 1.0\ \mathsf{else}\ 0.0$
$(\gg\!=) : \mathbf{Vec}\ A \to (A \to \mathbf{Vec}\ B) \to \mathbf{Vec}\ B$
$va \gg\!= f = \lambda b. \sum a\ (va\ a)(f\ a\ b)$
Auxiliary Definitions
$fun2lin : (A \to B) \to \mathbf{Lin}\ A\ B$
$fun2lin\ f = \lambda\ a.return\ (f\ a)$
$(\langle * \rangle) : \mathbf{Vec}\ A \to \mathbf{Vec}\ B \to \mathbf{Vec}\ (A, B)$
$v_1 \langle * \rangle v_2 = \lambda\ (a, b).v_1\ a\ *\ v_2\ b$
Arrow Types and Definitions
$arr : (A \to B) \to \mathbf{Super}\ A\ B$
$arr\ f = fun2lin\ (\lambda\ (b_1, b_2) \to (f\ b_1, f\ b_2))$
$(\ggg) :: (\mathbf{Super}\ A\ B) \to (\mathbf{Super}\ B\ C) \to (\mathbf{Super} A\ C)$
$f \ggg g = \lambda\ b.(f\ b \gg\!= g)$
$first :: (\mathbf{Super}\ A\ B) \to (\mathbf{Super}\ (A \times C)\ (B \times C))$
$first\ f\ ((b_1, d_1), (b_2, d_2)) = permute\ ((f(b_1, b_2)) \langle * \rangle\ return\ (d_1, d_2))$
$\quad \mathsf{where}\ permute\ v\ ((b_1, b_2), (d_1, d_2)) = v\ ((b_1, d_1), (b_2, d_2))$

Fig. 2. Quantum Arrows

to a density matrix of type B. This type can be understood by interpreting it in the same style as **Lin**.

We have defined in our previous work [5] the arrow operations for quantum computations into two levels. First we have proved that *pure* quantum states (i.e, vector states) are an instance of the concept of monads [7]. The definitions of the monadic functions are shown in Figure 2. The function *return* specifies how to construct vectors and $\gg\!=$ defines the behavior of an application of matrix to a vector. Moreover we have used the auxiliary functions *fun2lin*, which converts a classical (reversible) function to a linear operator, and $\langle * \rangle$ which is the usual tensor product in vector spaces.

The function *arr* constructs a quantum superoperator from a pure function by applying the function to both vector and its dual. The composition of arrows just composes two superoperators using the monadic *bind*. The function *first* applies the superoperator f to the first component (and its dual) and leaves the second component unchanged.

We have proved in our previous work that this superoperator instance of arrows satisfy the required nine equations [5].

4 The Arrow Calculus

In this section we present the arrow calculus [9] and show the translation of the calculus to classic arrows (described in Section 2) and vice versa. The translation is important because it essentially corresponds to the denotational semantic function for the quantum version of the arrow calculus. The material of this section closely follows the original presentation in [9].

4.1 The Calculus

The arrow calculus as shown in Figure 3 extends the core lambda calculus with four constructs satisfying five laws. Type $A \rightsquigarrow B$ denotes a computation that accepts a value of type A and returns a value of type B, possibly performing some side effects.

Syntax

Types	A, B, C	$::= \ldots \mid A \rightsquigarrow B$
Terms	L, M, N	$::= \ldots \mid \lambda^\bullet x.Q$
Commands	P, Q, R	$::= L \bullet P \mid [M] \mid \mathsf{let}\ x = P\ \mathsf{in}\ Q$

Types

$$\frac{\Gamma; x : A \vdash Q ! B}{\Gamma \vdash \lambda^\bullet x.Q : A \rightsquigarrow B} \qquad \frac{\Gamma \vdash L : A \rightsquigarrow B \quad \Gamma; \Delta \vdash M : A}{\Gamma; \Delta \vdash L \bullet M ! B}$$

$$\frac{\Gamma, \Delta \vdash M : A}{\Gamma; \Delta \vdash [M] ! A} \qquad \frac{\Gamma; \Delta \vdash P ! A \quad \Gamma; \Delta, x : A \vdash Q ! B}{\Gamma; \Delta \vdash \mathsf{let}\ x = P\ \mathsf{in}\ Q ! B}$$

Laws

$(\beta^\rightsquigarrow)$	$(\lambda^\bullet x.Q) \bullet M$	$= Q[x := M]$
$(\eta^\rightsquigarrow)$	$\lambda^\bullet x.(L \bullet [x])$	$= L$
(left)	$\mathsf{let}\ x = [M]\ \mathsf{in}\ Q$	$= Q[x := M]$
(right)	$\mathsf{let}\ x = P\ \mathsf{in}\ [x]$	$= P$
(assoc)	$\mathsf{let}\ y = (\mathsf{let}\ x = P\ \mathsf{in}\ Q)\ \mathsf{in}\ R$	$= \mathsf{let}\ x = P\ \mathsf{in}\ (\mathsf{let}\ y = Q\ \mathsf{in}\ R)$

Fig. 3. Arrow Calculus

There are two syntactic categories. Terms are ranged over by L, M, N, and commands are ranged over by P, Q, R. In addition to the terms of the core lambda calculus, there is one new term form: arrow abstraction $\lambda^\bullet x.Q$. There are three command forms: arrow application $L \bullet M$, arrow unit $[M]$ (which resembles unit in a monad), and arrow bind $\mathsf{let}\ x = P\ \mathsf{in}\ Q$ (which resembles bind in a monad).

In addition to the term typing judgment $\Gamma \vdash M : A$ there is also a command typing judgment $\Gamma; \Delta \vdash P ! A$. An important feature of the arrow calculus is that the command type judgment has two environments, Γ and Δ, where variables in Γ come from ordinary lambda abstractions $\lambda x.N$, while variables in Δ come from arrow abstraction $\lambda^\bullet x.Q$.

Arrow abstraction converts a command into a term. Arrow abstraction closely resembles function abstraction, save that the body Q is a command (rather than a term) and the bound variable x goes into the second environment (separated from the first by a semicolon).

Conversely, arrow application, $L \bullet M!B$ embeds a term into a command. Arrow application closely resembles function application. The arrow to be applied is denoted by a term, not a command; this is because there is no way to apply an arrow that is itself yielded. This is why there are two different environments, Γ and Δ: variables in Γ may denote arrows that are applied to arguments, but variables in Δ may not.

Arrow unit, $[M]!A$, promotes a term to a command. Note that in the hypothesis there is a term judgment with one environment (i.e, there is a comma between Γ and Δ), while in the conclusion there is a command judgment with two environments (i.e, there is a semicolon between Γ and Δ).

Lastly, using `let`, the value returned by a command may be bound.

Arrow abstraction and application satisfy beta and eta laws, ($\beta^{\rightsquigarrow}$) and ($\eta^{\rightsquigarrow}$), while arrow unit and bind satisfy left unit, right unit, and associativity laws, (left), (right), and (assoc). The beta law equates the application of an abstraction to a bind; substitution is not part of beta, but instead appears in the left unit law. The (assoc) law has the usual side condition, that x is not free in R.

4.2 Translation

The translation from the arrow calculus to classic arrows, shown below, gives a denotational semantics for the arrow calculus.

$$\begin{array}{ll}
[\![\lambda^\bullet x.Q]\!] & = [\![Q]\!]_x \\
[\![L \bullet M]\!]_\Delta & = arr(\lambda \Delta.[\![M]\!]) \ggg [\![L]\!] \\
[\![[M]]\!]_\Delta & = arr(\lambda \Delta.[\![M]\!]) \\
[\![\mathsf{let}\ x = P\ \mathsf{in}\ Q]\!]_\Delta & = (arr\ id \mathbin{\&\&\&} [\![P]\!]_\Delta) \ggg [\![Q]\!]_{\Delta,x}
\end{array}$$

An arrow calculus term judgment $\Gamma \vdash M : A$ maps into a classic arrow judgment $\Gamma \vdash [\![M]\!] : A$, while an arrow calculus command judgment $\Gamma; \Delta \vdash P!A$ maps into a classic arrow judgment $\Gamma \vdash [\![P]\!]_\Delta : \Delta \rightsquigarrow A$. Hence, the denotation of a command is an arrow, with arguments corresponding to the environment Δ and result of type A.

We omitted the translation of the constructs of core lambda calculus as they are straightforward homomorphisms. The translation of the arrow abstraction $\lambda^\bullet x.Q$ just undoes the abstraction and call the interpretation of Q using x. Application $L \bullet P$ translates to ¿, $[M]$ translates to arr and $\mathsf{let}\ x = P\ \mathsf{in}\ Q$ translates to pairing &&&(to extend the environment with P) and composition ¿(to then apply Q).

The inverse translation, from classic arrows to the arrow calculus is defined as:

$$\begin{array}{ll}
[\![arr]\!]^{-1} & = \lambda f.\lambda^\bullet x.[f\ x] \\
[\![(\ggg)]\!]^{-1} & = \lambda f.\lambda g.\lambda^\bullet x.g \bullet (f \bullet x) \\
[\![first]\!]^{-1} & = \lambda f.\lambda^\bullet z.\mathsf{let}\ x = f \bullet \mathsf{fst}\ z\ \mathsf{in}\ [(x, \mathsf{snd}\ z)]
\end{array}$$

Again we omitted the translation of the constructs of core lambda calculus as they are straightforward homomorphisms. Each of the three constants from classic arrows translates to an appropriate term in the arrow calculus.

5 The Arrow Calculus as a Quantum Programming Language

In this section we discuss how the arrow calculus can be used as a quantum programming language.

We start by showing quantum programs using the standard quantum circuit notation. The lines carry quantum bits. The values flow from left to right in steps corresponding to the alignment of the boxes which represent quantum gates. Gates connected via bullets to another wire are called *controlled operations*, that is, the wire with the bullet conditionally controls the application of the gate. The circuit in Figure 4 represents a quantum program for the Toffoli gate. Using the classic arrows approach for quantum programming presented in Section 3 and using the type of booleans, **Bool**, as the orthonormal basis for the qubit, this program would be codded as follows:

$$
\begin{aligned}
&\mathsf{toffoli} :: \mathbf{Super}\ (\mathbf{Bool}, \mathbf{Bool}, \mathbf{Bool})\ (\mathbf{Bool}, \mathbf{Bool}, \mathbf{Bool})\\
&\mathsf{toffoli} = arr\ (\lambda(a_0, b_0, c_0) \rightarrow (c_0, (a_0, b_0))) \ggg\\
&\qquad (first\ H \ggg arr\ (\lambda(c_1, (a_0, b_0)) \rightarrow ((b_0, c_1), a_0))) \ggg\\
&\qquad (first\ cV \ggg arr\ (\lambda((b_1, c_2), a_0) \rightarrow ((a_0, b_1), c_2))) \ggg\\
&\qquad (first\ cNot \ggg arr\ (\lambda((a_1, b_2), c_2) \rightarrow ((b_2, c_2), a_1))) \ggg \ldots
\end{aligned}
$$

As already noted by Paterson [11] this notation is cumbersome for programming. This is a "point-free" notation, rather different from the usual way of writing functional programs, with λ and let. Paterson introduced syntactic sugar for arrows, which we have used in our previous work [5]. However, the notation simply abbreviates terms built from the three constants, and there is no claim about reasoning with arrows. Using the *quantum arrow calculus* presented in Figure 5, this program would be like:

$$
\begin{aligned}
&\mathsf{toffoli} :: \mathbf{Super}\ (\mathbf{Bool}, \mathbf{Bool}, \mathbf{Bool})\ (\mathbf{Bool}, \mathbf{Bool}, \mathbf{Bool})\\
&\mathsf{toffoli} = \lambda^{\bullet}.(x, y, z).\mathsf{let}\ z' = H \bullet z\ \mathsf{in}\\
&\qquad \mathsf{let}\ (y', z'') = cV \bullet (y, z')\ \mathsf{in}\\
&\qquad \mathsf{let}\ (x', y'') = cNot \bullet (x, y')\mathsf{in} \ldots
\end{aligned}
$$

This style is more convenient and elegant as it is very similar to the usual familiar classical functional programming and is amenable to formal reasoning in

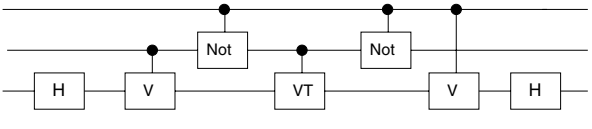


Fig. 4. Circuit for the Toffoli gate

a convenient way. Consider, for instance, the program which applies the quantum not gate twice. That is obviously equivalent to identity. To do such a simple proof using the *classic arrows* we need to learn how to use the *nine* arrow laws and also to recover the definitions of the functions *arr*, $\ggg$ and *first* for quantum computations presented in Figure 2.

The action of the quantum not gate, QNot, is to swap the amplitude probabilities of the qubit. For instance, QNot applied to $|0\rangle$ returns $|1\rangle$, and vice versa. But QNot applied to $\alpha|0\rangle + \beta|1\rangle$ returns $\alpha|1\rangle + \beta|0\rangle$.

Given the classical definition of not as follows:

$$\mathsf{not} = \lambda x.\mathsf{if}\ x == True\ \mathsf{then}\ False\ \mathsf{else}\ True : \ \mathbf{Bool} \to \mathbf{Bool}$$

Using the arrow calculus, the QNot would be written as:

$$\mathsf{QNot} = \lambda^{\bullet} y.[\mathsf{not}\ y] : \ \mathbf{Super\ Bool\ Bool}.$$

Then, the program which applies the QNot twice, would be:

$$\Gamma \vdash \lambda^{\bullet} x.\mathsf{let}\ w = (\lambda^{\bullet} z.[\mathsf{not}\ z]) \bullet x\ \mathsf{in}\ (\lambda^{\bullet} y.[\mathsf{not}\ y]) \bullet w$$

Again the syntax, with arrow abstraction and application, resembles lambda calculus. Now we can use the intuitive arrow calculus laws (from Figure 3) to prove the obvious equivalence of this program with identity. The proof follows the same style of the proofs in classical functional programming.

$$\begin{array}{ll}
\lambda^{\bullet} x.\mathsf{let}\ w = (\lambda^{\bullet} z.[\mathsf{not}\ z]) \bullet x\ \mathsf{in}\ (\lambda^{\bullet} y.[\mathsf{not}\ y]) \bullet w & =^{(\beta^{\rightsquigarrow})} \\
\lambda^{\bullet} x.\mathsf{let}\ w = [\mathsf{not}\ x]\ \mathsf{in}\ (\lambda^{\bullet} y.[\mathsf{not}\ y]) \bullet w & =^{(\mathrm{left})} \\
\lambda^{\bullet} x.(\lambda^{\bullet} y.[\mathsf{not}\ y]) \bullet (\mathsf{not}\ x) & =^{(\beta^{\rightsquigarrow})} \\
\lambda^{\bullet} x.[\mathsf{not}(\mathsf{not}\ x)] & =^{def.not} \\
\lambda^{\bullet} x.[x] &
\end{array}$$

It is interesting to note that we have two ways for defining superoperators. The first way is going directly from classical functions to superoperators as we did above for *not*, using the default definition of *arr*. The other way is going from the monadic pure quantum functions to superoperators. As monads are a special case of arrows [6] there is *always* a translation from monadic functions to arrows. Hence, any **Lin** A B is a special case of **Super** A B.

Hence, we construct the quantum arrow calculus in Figure 5 in three levels. First we inherit all the constructions from simply-typed lambda calculus with the type of booleans and with classical let and if (see Appendix A). Then we add the monadic *unit*, [ ], to build pure vectors (over booleans), let to sequence computations with vectors, and plus and minus to add and subtract vectors (the monadic calculus [7] with its laws is presented in Appendix B). Finally, we add the constructions of the arrow calculus. The appeal of using the arrows approach is because we can express measurement operations (i.e, extract classical information from the quantum system) inside the formalism. Therefore, we have two computations for measurements on mixed states, meas and trL. The computation meas returns a classical value and a post-measurement state of the quantum

Syntax

Types $A, B, C ::= \ldots \mid \mathbf{Bool} \mid \mathbf{Dens}\ A \mid \mathbf{Vec}\ A \mid \mathbf{Super}\ A\ B$

Terms $L, M, N ::= [T] \mid \mathsf{let}\ x = M\ \mathsf{in}\ N \mid \lambda^{\bullet} x.Q \mid + \mid -$

Commands $P, Q, R ::= L \bullet P \mid [M] \mid \mathsf{let}\ x = P\ \mathsf{in}\ Q \mid \mathsf{meas} \mid \mathsf{trL}$

Monad Types

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash [M] : \mathbf{Vec}\ A} \qquad \frac{\Gamma \vdash M : \mathbf{Vec}\ A \quad \Gamma, x : A \vdash N : \mathbf{Vec}\ B}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : \mathbf{Vec}\ B}$$

$$\frac{\Gamma \vdash M, N : \mathbf{Vec}\ A}{\Gamma \vdash M + N : \mathbf{Vec}\ A} \qquad \frac{\Gamma \vdash M, N : \mathbf{Vec}\ A}{\Gamma \vdash M - N : \mathbf{Vec}\ A}$$

Arrow Types

$$\frac{\Gamma; x : A \vdash Q ! \ \mathbf{Dens}\ B}{\Gamma \vdash \lambda^{\bullet} x.Q : \mathbf{Super}\ A\ B} \qquad \frac{\Gamma \vdash L : \mathbf{Super}\ A\ B \quad \Gamma; \Delta \vdash M :\ A}{\Gamma; \Delta \vdash L \bullet M ! \ \mathbf{Dens}\ B}$$

$$\frac{\Gamma, \Delta \vdash M : A}{\Gamma; \Delta \vdash [M] ! \ \mathbf{Dens}\ A} \qquad \frac{\Gamma; \Delta \vdash P ! \ \mathbf{Dens}\ A \quad \Gamma; \Delta, x : A \vdash Q ! \ \mathbf{Dens}\ B}{\Gamma; \Delta \vdash \mathsf{let}\ x = P\ \mathsf{in}\ Q ! \ \mathbf{Dens}\ B}$$

$$\frac{}{\Gamma; x : A \vdash \mathsf{meas} \ ! \ \mathbf{Dens}\ (A, A)} \qquad \frac{}{\Gamma; x : (A, B) \vdash \mathsf{trL} \ ! \ \mathbf{Dens}\ B}$$

Fig. 5. Quantum Arrow Calculus

system. The computation trL *traces out* or *projects* part of the quantum state (the denotation of these operations is provided in Appendix D).

To exemplify the use of the monadic constructions, consider, for example, the hadamard quantum gate, which is the source of superpositions. For instance, hadamard applied to $|0\rangle$ returns $|0\rangle + |1\rangle$, and applied to $|1\rangle$ returns $|0\rangle - |1\rangle$. But, hadamard applied to $|0\rangle + |1\rangle$ returns $|0\rangle$, as it is a reversible gate. To define this program in the quantum arrow calculus, we just need to define its work for the basic values, $|0\rangle$ and $|1\rangle$, as follows:

$$\begin{aligned} \mathsf{hadamard} = \lambda x.\mathsf{if}\ x == True\ \ &\mathsf{then}\ [False] - [True] \\ &\mathsf{else}\ [False] + [True] : \mathbf{Lin\ Bool\ Bool} \end{aligned}$$

Then, the superoperator would be:

$$\mathsf{Had} = \lambda^{\bullet} y.[\mathsf{hadamard}\ y] : \mathbf{Super\ Bool\ Bool}$$

Another interesting class of operations are the so-called *quantum controlled operations*. For instance, the controlled not, Cnot, receives two qubits and applies a not operation on the second qubit depending on the value of the first qubit. Again, we just need to define it for the basic quantum values:

$$\begin{aligned} \mathsf{cnot} = \lambda(x, y).\mathsf{if}\ x\ \ &\mathsf{then}\ [(x, \mathsf{not}\ y)] \\ &\mathsf{else}\ [(x, y)] : \mathbf{Lin}\ (\mathbf{Bool}, \mathbf{Bool})\ (\mathbf{Bool}, \mathbf{Bool}) \end{aligned}$$

Again, the superoperator of type **Super** (**Bool**, **Bool**) (**Bool**, **Bool**) would be $\mathsf{Cnot} = \lambda^\bullet(x, y).[\mathsf{cnot}\ (x, y)]$.

The motivation of using superoperators is that we can express *measurement* operations inside of the formalism. One classical example of quantum algorithm which requires a measurement operation is the quantum teleportation [4]. It allows the transmission of a qubit to a partner with whom is shared an entangled pair. Below we define the two partners of a teleportation algorithm.

$$\begin{aligned}
&\mathsf{Alice} : \mathbf{Super}\ (\mathbf{Bool}, \mathbf{Bool})\ (\mathbf{Bool}, \mathbf{Bool}) \\
&\mathsf{Alice} = \lambda^\bullet(x, y).\ \mathsf{let}\ (x', y') = \mathsf{Cnot} \bullet (x, y)\ \mathsf{in} \\
&\qquad\qquad \mathsf{let}\ q = (\mathsf{Had} \bullet x', y')\ \mathsf{in} \\
&\qquad\qquad\quad \mathsf{let}\ (q', v) = \mathsf{meas} \bullet q\ \mathsf{in}\ \mathsf{trL} \bullet (q, v)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{Bob} : \mathbf{Super}\ (\mathbf{Bool}, \mathbf{Bool}, \mathbf{Bool})\ \mathbf{Bool} \\
&\mathsf{Bob} = \lambda^\bullet(x, y, z).\ \mathsf{let}\ (z', x') = \mathsf{Cnot} \bullet (z, x)\ \mathsf{in} \\
&\qquad\qquad \mathsf{let}\ (y', x'') = (\mathsf{Cz} \bullet (y, x'))\ \mathsf{in}\ \mathsf{trL} \bullet ((y', z'), x'')
\end{aligned}$$

6 Conclusion

We have presented a lambda calculus for general quantum programming that builds on well-understood and familiar programming patterns and reasoning techniques. Besides supporting an elegant functional programming style for quantum computations, the quantum arrow calculus allows reasoning about *general* or *mixed* quantum computations. This is the first work proposing reasoning about *mixed* quantum computations. The equations of the arrow calculus plus the equations of the monadic calculus provide indeed a powerful mechanism to make proofs about quantum programs. In [12] we have proposed very similar reasoning techniques, however for *pure* quantum programs. Also, in [13] the author presents a quantum lambda calculus based on linear logic, but just for pure quantum computations.

Acknowledgements

We thank Jeremy Yallop for very helpful comments.

References

1. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
2. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proc. IEEE Symposium on Foundations of Computer Science, pp. 124–134 (1994)
3. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proc. 28. Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)

4. Bennett, C.H., Brassard, G., Crepeau, C., Jozsa, R., Peres, A., Wootters, W.: Teleporting an unknown quantum state via dual classical and EPR channels. Phys. Rev. Lett., 1895–1899 (1993)
5. Vizzotto, J.K., Altenkirch, T., Sabry, A.: Structuring quantum effects: Superoperators as arrows. Journal of Mathematical Structures in Computer Science: special issue in quantum programming languages 16, 453–468 (2006)
6. Hughes, J.: Generalising monads to arrows. Science of Computer Programming 37, 67–111 (2000)
7. Moggi, E.: Computational lambda-calculus and monads. In: Proceedings of the Fourth Annual Symposium on Logic in computer science, pp. 14–23. IEEE Press, Los Alamitos (1989)
8. Mcbride, C., Paterson, R.: Applicative programming with effects. J. Funct. Program. 18(1), 1–13 (2008)
9. Lindley, S., Wadler, P., Yallop, J.: The arrow calculus (functional pearl). In: International Conference on Functional Programming (2008)
10. Moggi, E.: Notions of computation and monads. Information and Computation 93(1), 55–92 (1991)
11. Paterson, R.: A new notation for arrows. In: Proc. International Conference on Functional Programming, pp. 229–240 (September 2001)
12. Altenkirch, T., Grattage, J., Vizzotto, J.K., Sabry, A.: An algebra of pure quantum programming. Electron. Notes Theor. Comput. Sci. 170, 23–47 (2007)
13. Tonder, A.v.: A lambda calculus for quantum computation. SIAM J. Comput. 33(5), 1109–1135 (2004)
14. MonadPlus (2005), http://www.haskell.org/hawiki/MonadPlus
15. Hinze, R.: Deriving backtracking monad transformers. In: ICFP 2000: Proceedings of the 5th ACM SIGPLAN International Conference on Functional Programming, pp. 186–197. ACM Press, New York (2000)
16. Aharonov, D., Kitaev, A., Nisan, N.: Quantum circuits with mixed states. In: Proceedings of the thirteenth annual ACM symposium on Theory of computing, pp. 20–30. ACM Press, New York (1998)
17. Selinger, P.: Towards a quantum programming language. Journal of Mathematical Structures in Computer Science: special issue in quantum programming languages 16, 527–586 (2006)

A Simply-Typed Lambda Calculus

The simply-typed lambda calculus with the type of booleans, and with let and if is shown in Figure 6. Let A, B, C range over types, L, M, N range over terms, and Γ, Δ range over environments. A type judgment $\Gamma \vdash M : A$ indicates that in environment Γ term M has type A. As presented in the arrow calculus [9], we are using a Curry formulation, eliding types from terms.

B Monadic Calculus

The simply-typed lambda calculus presented in Appendix A is the foundation of purely functional programming languages. In this section we show the *monadic calculus* [7], which also models monadic effects. A monad is represented using a type constructor for computations m and two functions: $return :: a \to m\ a$

Syntax

Types $A, B, C ::= \mathbf{Bool} \mid A \times B \mid A \to B$

Terms $L, M, N ::= x \mid \mathsf{True} \mid \mathsf{False} \mid (M, N) \mid \mathsf{fst}\ L \mid \mathsf{snd}\ L \mid \lambda x.N \mid L\ M$
$\mid \mathsf{let}\ x = M\ \mathsf{in}\ N \mid \mathsf{if}\ L\ \mathsf{then}\ M\ \mathsf{else}\ N$

Environments $\Gamma, \Delta ::= x_1 : A_1, \ldots, x_n : A_n$

Types

$$\frac{}{\emptyset \vdash \mathsf{False} : \mathbf{Bool}} \qquad \frac{}{\emptyset \vdash \mathsf{True} : \mathbf{Bool}} \qquad \frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B} \qquad \frac{\Gamma \vdash L : A \times B}{\Gamma \vdash \mathsf{fst}\ L : A} \qquad \frac{\Gamma \vdash L : A \times B}{\Gamma \vdash \mathsf{snd}\ L : B}$$

$$\frac{\Gamma, x : A \vdash N : B}{\Gamma \vdash \lambda x.N : A \to B} \qquad \frac{\Gamma \vdash L : A \to B \quad \Gamma \vdash M : A}{\Gamma \vdash L\ M : B}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : B} \qquad \frac{\Gamma \vdash L : \mathbf{Bool} \quad \Gamma \vdash M, N : B}{\Gamma \vdash\ \mathsf{if}\ L\ \mathsf{then}\ M\ \mathsf{else}\ N : B}$$

Laws

(β_1^x)	$\mathsf{fst}\ (M, N)$	$= M$
(β_2^x)	$\mathsf{snd}\ (M, N)$	$= N$
(η^x)	$(\mathsf{fst}\ L, \mathsf{snd} L)$	$= L$
$(\beta^{\to})$	$(\lambda x.N)M$	$= N[x := M]$
$(\eta^{\to})$	$\lambda x.(L\ x)$	$= L$
(let)	$\mathsf{let}\ x = M\ \mathsf{in}\ N$	$= N[x := M]$
(β_1^{if})	$\mathsf{if\ True\ then}\ M\ \mathsf{else}\ N$	$= M$
(β_2^{if})	$\mathsf{if\ False\ then}\ M\ \mathsf{else}\ N$	$= N$

Fig. 6. Simply-typed Lambda Calculus

and $\gg\!=\!:: m\ a \to (a \to m\ b) \to m\ b$. The operation $\gg\!=$ (pronounced "bind") specifies how to sequence computations and *return* specifies how to lift values to computations. From a programming perspective, a *monad* is a construct to structure *computations*, in a functional environment, in terms of values and sequence of computations using those values.

The monadic calculus extends the simply-typed lambda calculus with the constructs in Figure 7. Unit and bind satisfy left unit, right unit, and associativity laws, (left), (right), and (assoc).

Beyond the three monad laws discussed above, some monads obey the MonadPlus laws. The MonadPlus interface provides two primitives, mzero and + (called mplus), for expressing choices. The command + introduces a choice junction, and mzero denotes failure.

The precise set of laws that a `MonadPlus` implementation should satisfy is not agreed upon [14], but in [15] is presented a reasonable agreement on the laws. We use in Figure 7 the laws introduced by [15].

The intuition behind these laws is that MonadPlus is a disjunction of goals and $\gg\!=$ is a conjunction of goals. The conjunction evaluates the goals from left-to-right and is not symmetric.

Syntax

Types $A, B, C ::= ... \mid \mathbf{M}\ A$

Terms $L, M, N ::= ... \mid [M] \mid \mathsf{let}\ x = M\ \mathsf{in}\ N \mid \mathsf{mzero} \mid + \mid -$

Monadic Types

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash [M] : \mathbf{M}\ A} \qquad \frac{\Gamma \vdash M : \mathbf{M}\ A \quad \Gamma, x : A \vdash N : \mathbf{M}\ B}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : \mathbf{M}\ B}$$

MonadPlus Types

$$\frac{}{\Gamma \vdash \mathsf{mzero} : \mathbf{M}\ A} \qquad \frac{\Gamma \vdash M, N : \mathbf{M}\ A}{\Gamma \vdash M + N : \mathbf{M}\ A}$$

Laws

(left) $\mathsf{let}\ x = [L]\ \mathsf{in}\ N = N[x := L]$

(right) $\mathsf{let}\ x = L\ \mathsf{in}\ [x] = L$

(assoc) $\mathsf{let}\ y = (\mathsf{let}\ x = L\ \mathsf{in}\ N)\ \mathsf{in}\ T = \mathsf{let}\ x = L\ \mathsf{in}\ (\mathsf{let}\ y = N\ \mathsf{in}\ T)$

MonadPlus Laws

$\mathsf{mzero} + a = a$

$a + \mathsf{mzero} = a$

$a + (b + c) = (a + b) + c$

$\mathsf{let}\ x = \mathsf{mzero}\ \mathsf{in}\ T = \mathsf{mzero}$

$\mathsf{let}\ x = (M + N)\ \mathsf{in}\ T = (\mathsf{let}\ x = M\ \mathsf{in}\ T) + (\mathsf{let}\ x = N\ \mathsf{in}\ T)$

Fig. 7. Monadic Calculus

C General Quantum Computations

Quantum computation, as its classical counterpart, can be seen as processing of information using quantum systems. Its basic idea is to encode data using quantum bits (qubits). In quantum theory, considering a *closed* quantum system, the qubit is a *unit* vector living in a complex inner product vector space know as *Hilbert space* [1]. We call such a vector a *ket* (from *Dirac's notation*) and denote it by $|v\rangle$ (where v stands for elements of an orthonormal basis), a column vector. Differently from the classical bit, the qubit can be in a *superposition* of the two basic states written as $\alpha|0\rangle + \beta|1\rangle$, or

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

with $|\alpha|^2 + |\beta|^2 = 1$. Intuitively, one can think that a qubit can exist as a 0, a 1, or simultaneously as both 0 and 1, with numerical coefficient (i.e., the probability amplitudes α and β) which determines the probability of each state. The quantum superposition phenomena is responsible for the so called "quantum parallelism."

Operations acting on those *isolated* or *pure* quantum states are linear operations, more specifically *unitary matrices* S. A matrix A is called *unitary* if $S^*S = I$, where S^* is the adjoint of S, and I is the identity. Essentially, those unitary transformations act on the quantum states by changing their probability

amplitudes, without loss of information (i.e., they are reversible). The application of a unitary transformation to a state vector is given by usual matrix multiplication.

Unfortunately in this model of quantum computing, it is difficult or impossible to deal formally with another class of quantum effects, including measurements, decoherence, or noise.

Measurements are critical to some quantum algorithms, as they are the only way to extract *classical* information from quantum states.

A *measurement* operation projects a quantum state like $\alpha|0\rangle + \beta|1\rangle$ onto the basis $|0\rangle$,$|1\rangle$. The outcome of the measurement is not deterministic and it is given by the probability amplitude, i.e., the probability that the state after the measurement is $|0\rangle$ is $|\alpha|^2$ and the probability that the state is $|1\rangle$ is $|\beta|^2$. If the value of the qubit is initially unknown, than there is no way to determine α and β with that single measurement, as the measurement may *disturb* the state. But, *after* the measurement, the qubit is in a *known* state; either $|0\rangle$ or $|1\rangle$. In fact, the situation is even more complicated: measuring part of a quantum state collapses not only the measured part but any other part of the global state with which it is *entangled*. In an entangled state, two or more qubits have to be described with reference to each other, even though the individuals may be spatially separated [1].

There are several ways to deal with measurements in quantum computing, as summarized in our previous work [5]. To deal formally and elegantly with measurements, the state of the computation is represented using a *density matrix* and the operations are represented using *superoperators* [16]. Using these notions, the *projections* necessary to express measurements become expressible within the model.

Intuitively, density matrices can be understood as a statistical perspective of the state vector. In the density matrix formalism, a quantum state that used to be modeled by a vector $|v\rangle$ is now modeled by its outer product $|v\rangle\langle v|$, where $\langle v|$ is the row vector representing the adjoint (or dual) of $|v\rangle$. For instance, the state of a quantum bit $|v\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ is represented by the density matrix:

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Note that the main diagonal shows the classical probability distribution of basic quantum states, that is, these state has $\frac{1}{2}$ of probability to be $|0\rangle$ and $\frac{1}{2}$ of probability to be $|1\rangle$.

However, the appeal of density matrices is that they can represent states other than the pure ones above. In particular if we perform a measurement on the state represented above, we should get $|0\rangle$ with probability 1/2 or $|1\rangle$ with probability 1/2. This information, which cannot be expressed using vectors, can be represented by the following density matrix:

$$\begin{pmatrix} 1/2 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1/2 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

[1] For more detailed explanation about entangled, see [1].

Such a density matrix represents a *mixed state* which corresponds to the sum (and then normalization) of the density matrices for the two results of the observation.

The two kinds of quantum operations, namely unitary transformation and measurement, can both be expressed with respect to density matrices [17]. Those operations now mapping density matrices to density matrices are called *super-operators*. A unitary transformation S maps a pure quantum state $|u\rangle$ to $S|u\rangle$. Thus, it maps a pure density matrix $|u\rangle\langle u|$ to $S|u\rangle\langle u|S^*$. Moreover, a unitary transformation extends linearly to mixed states, and thus, it takes any mixed density matrix A to SAS^*.

As one can observe in the resulting matrix above, to execute a measurement corresponds to setting a certain region of the input density matrix to zero.

D Definition of Measurement Operations

In this section we present the denotations of the programs for measurements, *trl* and *meas*, added to the quantum arrow calculus.

$$\mathsf{trL} :: \mathbf{Super}\ (A, B)\ B$$
$$\mathsf{trL}((a_1, b_1), (a_2, b_2)) = \mathsf{if}\ a_1 == a_2\ \mathsf{then}\ return(b_1, b_2)\ \mathsf{else}\ \mathsf{mzero}$$

$$\mathsf{meas} :: \mathbf{Super}\ A\ (A, A)$$
$$\mathsf{meas}(a_1, a_2) = \mathsf{if}\ a_1 == a_2\ \mathsf{then}\ return((a_1, a_1), (a_1, a_1))\ \mathsf{else}\ \mathsf{mzero}$$

We consider *projective* measurements which are described by a set of projections onto mutually orthogonal subspaces. This kind of measurement returns a classical value and a post-measurement state of the quantum system. The operation *meas* is defined in such a way that it can encompass both results. Using the fact that a classical value m can be represented by the density matrix $|m\rangle\langle m|$ the superoperator *meas* returns the output of the measurement attached to the post-measurement state.

Knowledge, Time, and Logical Omniscience

Ren-June Wang

Computer Science
CUNY Graduate Center
365 Fifth Avenue, New York, NY 10016
rwang@gc.cuny.edu

Abstract. Knowledge's acquisition happens *in* time. However, this feature is not reflected in the standard epistemic logics, e.g. S4 with its possible world semantics suggested by Hintikka in [1], and hence their applications are limited. In this paper we adapt these normal modal logics to increase their expressive power such that not only is *what* is known modeled but also *when* it is known is recorded. We supplement each world with an awareness function which is an augmentation of Fagin-Halpern's to keep track of the time when each formula is to be derived. This provides a new response to the logical omniscience problem. Our work originates from the tradition of study of *Justification Logic*, also known as *Logic of Proofs*, LP, introduced by Artemov ([2],[3],[4]). We will give the axiom systems of the models built here, accompanied with soundness and completeness results.

1 Introduction

To acquire new knowledge, we think, ponder, infer, and so on. All these activities take time. However, this feature of knowledge is not reflected in the standard epistemic logic such as S4, and hence their applications are limited. Since there's no time components helping to indicate when knowledge is acquired by the knower, all knowledge, which is closed under logical consequence, has to be assumed to be known at the one time. This makes the model unrealistic. Our knower is *logical omniscience*.

There has been many approaches advocated in the literature to solve this problem. Based on the understanding that the knower modeled by modal epistemic logics knows too much, these approaches provides various mechanisms such as impossible worlds ([5]), awareness functions ([6]), and deduction structures with incomplete deduction rules ([7]), to name some of them, to weaken standard epistemic logics.[1]. Also in [10] the logical omniscience problem is understood as a proof complexity problem. It is suggested that an epistemic logic system is logical omniscience if and only if some knowledge assertion in the system cannot be supported by a feasible size proof. And then it is shown that the *Logic of Proofs*, LP ([3], [4]), now understood as Justification Logic ([2]), is not logical omniscience.

[1] Cf. [8] and chapter 9 in [9] for a survey of the logical omniscience problem.

H. Ono, M. Kanazawa, and R. de Queiroz (Eds.): WoLLIC 2009, LNAI 5514, pp. 394–407, 2009.

To solve this problem, however, we choose to adapt the semantics for these standard epistemic logics such that not only will what is known be modeled but also when it is to be known will be recorded. In our point of view, it is not so much a problem for an epistemic logic system which is intended to model an intelligent agents, who are capable of performing logical inferences, to be logical omniscience. But to enhance the applicability, the system has to be accommodated to express the time, or other resources, needed for the knowledge acquisition such that the evolution of knowledge over time can be grasped. For this purpose, we introduce a collection of $\mathtt{S4}^{\Delta}$ logics, in which formulas such as $\mathbf{K}F^i$ are included with i a natural number, meaning formula F is known at time i. The reason why we can have not a single one but a collection of logics corresponding to S4, is that our method is flexible enough to model knowers with different initial knowledges. Some of these logics with richer initial knowledges have been proved having the desired result, the *realization theorem*: every S4 theorem is a theorem in these logics without number labels.

In the literature, there have been many logics proposed combining epistemic and temporal modals, and they have proved very useful.[2] In the semantics of these logics, the time components that the temporal modals range over are better understood as rounds, stages, phases, etc. In each round, the knowers have new information, and their knowledge gets updated. However, the epistemic modals in these logics are essentially modals in normal modal logics, and hence knowers know all the logical consequences of their knowledge within each round. Our approach differs from these in not only using the method of explicit time points, other than temporal modals, to deal with time, but also focusing on increasing the expressive power of standard epistemic logics such that when these logics are applied, we are able to decide, for example, in a round, what can be known by a realistic knower, when whatever time bound is given for that round.

The method we will employ here is basically to extend the use of awareness function, a concept introduced by Fagin and Halpern in [6] to give a possible solution to the logical omniscience problem. In Fagin-Halpern approach, to say that the knower knows some formula at a world means not only is the formula true at all epistemic alternatives to the world, but also the formula should be aware of at the world. In the same paper, they also suggested the possibility to utilize awareness functions so that time can be put into the picture. Our method can be viewed as a direct response to this suggestion. In our usage, not only is what formulas to be aware of in each world provided, but also when will these formulas to be aware of is given. More details will be in the sequel.

The initial of our paper, however, comes from the tradition of study of logics of justification, which began with the *Logic of Proofs*, LP, introduced by Artemov as an explicit proof counterpart of the modal logic S4. The axiom system $\mathtt{S4}^{\Delta}$ of the semantics we are going to present here was first introduced in [17] as an intermediate logic for to discuss the relations between proofs in S4 and proofs in LP. A syntactical proof of the *realization theorem* has been given there. Our model is adapted from Fitting's work [18] on the Kripke-style semantics for

[2] See [11], [12], [13], [14], [15], [16], and also chapters 4 and 8 in [9].

LP, and some terminology is borrowed from there. Except for those necessary alternation from objects representing justifications to time points, we also make modifications both to fit the intuition concerning dealing with time points and for comparisons with other neighboring logics of S4. More words about comparisons between S4 and $\mathtt{S4}^{\Delta}$ and between $\mathtt{S4}^{\Delta}$ and LP will be said later in the paper.

In the final section, we will consider the Δ-style counterparts of other normal modal logics, including logical system with the 5 axiom.

2 Semantics

We first review the possible world semantics in general and the semantics for S4 in particular. The language $L_{\mathbf{K}}$ for S4 and other standard epistemic logics is a extension of the language of propositional logic, i.e., formulas are built up from propositional letters and connectives $\neg$, $\rightarrow$, $\vee$ and $\wedge$, and it has an additional formula formation rule: if $F \in L_{\mathbf{K}}$ then $(\mathbf{K}F)$ is also in $L_{\mathbf{K}}$ (parentheses usually omitted), where $\mathbf{K}F$ means the knower knows F.

A frame is a structure $\langle W, R\rangle$, where W is a non-empty set of possible worlds, and R is a binary relation on W. A standard epistemic model is a structure $\mathcal{M} = \langle W, R, \mathcal{V}\rangle$, where $\langle W, R\rangle$ is a frame, and $\mathcal{V}$ is a valuation from propositional letters to worlds. Then the truth of formulas is defined as (for saving space, we omit the cases for $\wedge$ and $\vee$; nevertheless, they can be redefined by other connectives):

1. $(M, w) \Vdash P$ iff $w \in \mathcal{V}(P)$ for P is a propositional letter;
2. $(M, w) \Vdash \neg F$ iff $w \nVdash F$;
3. $(M, w) \Vdash F \rightarrow G$ iff $w \nVdash F$ or $w \Vdash G$;
4. $(M, w) \Vdash \mathbf{K}F$ iff $w' \Vdash F$ for all $w' \in W$ with wRw'.

In this semantics, the diversity of the knower's epistemic ability is reflected by imposing different conditions on the binary relation. In this paper we mainly consider the semantics which models the knowledge, contrary to the belief, of the knower who has the ability to introspect his/her own knowledge. Then we need the binary relation to be reflexive and transitive. This is the semantics of S4 suggested by Hintikka as an epistemic logic.

This possible world semantic is based on the motto "information is elimination of uncertainty." Even though this motto is quite intuitive, once we directly apply the semantics in a realistic environment, where the knower is supposed to take time to reason, and only a finite amount of time is allowed, our knower knows all logical truth within the amount of time. Most of complaint about this semantics is that the knower modeled knows too much. We, however, consider that there's nothing wrong to suppose that it is possible for the knower to know all these logical truths. But the problem is that different logical truth should take different amount of time to derive, given the knower having some basic logical truths and being able to apply some inference rules. This is just what we are going to do in the following to reveal the hidden temporal relations between knowledge in this semantics.

2.1 Δ-Semantics

The language L_Δ for our $\mathtt{S4}^\Delta$ logics and other Δ-logics is an extension of the language of propositional logic with the formula formation rule: if $F \in L_\Delta$ then $(\mathbf{K}F^i)$ is also in L_Δ, where i is a natural number. The intended meaning of formula $\mathbf{K}F^i$ is that the knower knows F at time i, or the formula F is known at time i (by the knower). We are intendedly not to write the formula as $\mathbf{K}^iF$ because it looks like presupposing that for each i, there is an independent knowledge operator $\mathbf{K}^i$.

The novel tool in our semantics is the awareness function α, which is a partial function mapping formulas in L_Δ to natural numbers. The purpose of the awareness function is to capture the knower's reasoning process by putting formulas in order. One formula is to be aware of later than another because it takes more time to be derived. Some conditions will be imposed on these functions later on to reflect the knower's reasoning ability. A Δ-model is a structure $M = \langle W, R, \{\alpha_w\}, \mathcal{V}\rangle$, where $\langle W, R, \mathcal{V}\rangle$ is a standard epistemic model and $\{\alpha_w\}$ is a collection of awareness functions with index $w \in W$. The truth of formulas in L_Δ is defined as above with the forth condition replaced by:

4'. $(M, w) \Vdash \mathbf{K}F^i$ iff
- $w' \Vdash F$ for all $w' \in W$ with wRw', and
- $\alpha_w(F) \leq i$

This condition says that the knower knows F at time i at a world w only if the formula is true at all epistemic alternatives of w and s/he is aware of the formula F before or at i.

2.2 $\mathtt{S4}^\Delta$-Awareness Functions

Awareness functions are grouped by their bases. An awareness base is a tuple $\mathcal{A} = \langle \mathbf{A}, f\rangle$, where the base set $\mathbf{A}$ is a collection of L_Δ formulas, and the base function f is a total function from $\mathbf{A}$ to natural numbers. Formulas in the base set are those to be aware of by the knower either from outside where someone tells him, or from inside where we assume the knower has them inherently, and the base function tells us when the knower is aware of these formulas.

There are three sets of rules that an awareness function should follow. First, given an awareness base $\mathcal{A}$, for an awareness function α based on $\mathcal{A}$ and a formula $A \in \mathcal{A}$, $\alpha(A) \leq f(A)$, i.e., the knower is aware of A before or at $f(A)$ since after $f(A)$ the knower must be aware of the formula. Second, we suppose that our knower has the basic reasoning strength: s/he can do modus ponens and for those formulas in the base set the knower is able to be aware of that s/he knows those formulas. The third set of rules aims to reflect epistemic ability which we assume the knower to possess. Here we need the rule that whatever the formulas that the knower is aware of, s/he is able to be aware of that s/he knows the formula. Notice that in our usage of being aware of a formula at time i only means that at the moment the knower is aware of the possibility of the formula

to be true. It does not mean that the formula must be true. It is possible for the knower to be aware of mutual contradictory formulas and eventually aware of all formulas. Here's the formal definition.

Definition 1. *Given an awareness base $\mathcal{A} = \langle \mathbf{A}, f \rangle$, an* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-awareness function *is a partial function from L_Δ to $\mathbb{N}$ satisfying the following conditions ($\alpha(A)\downarrow$ denotes $\alpha(A)$ is defined.):*

1. *Initial condition*
   - *if $A \in \mathbf{A}$, then $\alpha(A)\downarrow$ and $\alpha(A) \leq f(A)$,*
2. *Awareness by deduction*
   a. *if $\alpha(F \to G)\downarrow$ and $\alpha(F)\downarrow$, then*
      - $\alpha(G) \leq \max(\alpha(F \to G), \alpha(F)) + 1$,
   b. *if $A \in \mathbf{A}$ and $f(A) \leq i$, then*
      - $\alpha(\mathbf{K}A^i) \leq i + 1$,
3. ***Inner positive introspection***
   a. *if $F \in L_\Delta$ and $\alpha(F) \leq i$, then*
      - $\alpha(\mathbf{K}F^i) \leq i + 1$.

The condition 2b. is covered by the condition of inner positive introspection. One reason to separate them is that the former condition is more basic than the latter. The other reason is partially syntactical. We will discuss this more in the next section.

In our definition only general epistemic aspects of the knower are reflected in awareness functions. There's no position for the logical strength of the knower, that is, the ability to manipulate logical constants such as *and*, $\wedge$, and *or*, $\vee$. Alternatively, it is determined by the bases. For example, to say a knower knows the logical constant *and*, $\wedge$, is to say that formulas of these kinds, $F \to (G \to (F \wedge G))$ and $(F \wedge G) \to F$, $(F \wedge G) \to F$, are in the base.

Among all $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-awareness functions, there is one playing a special role.

Definition 2. *Given an awareness base $\mathcal{A}$, we say an* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$*-awareness function $\alpha^*_{\mathcal{A}}$ is* critical *if for any* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$*-awareness function α, and any $F \in L_\Delta$ with $\alpha^*_{\mathcal{A}}(F)\downarrow$, $\alpha(F) \leq \alpha^*_{\mathcal{A}}(F)$.*

Lemma 1. *For each awareness base $\mathcal{A}$, there exists a unique critical* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$*-awareness function $\alpha^*_{\mathcal{A}}$.*

2.3 $\mathtt{S4}^{\Delta}$-Semantics and $\mathtt{S4}^{\Delta}$-Awareness Bases

Definition 3. *Given an awareness base $\mathcal{A}$, a Δ-model $M = \langle W, R, \{\alpha_w\}, \mathcal{V} \rangle$ is an* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-model *if the frame $\langle W, R \rangle$ is reflexive and transitive, and $\{\alpha_w\}$ is a collection of* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$*-awareness functions and satisfies the* ***monotonicity*** *condition: for any $w\mathcal{R}w'$, $\alpha_{w'}(F) \leq \alpha_w(F)$.*

We say a formula is $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid in a $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-model if it is true at all worlds of the model, and a formula is $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid, denoted as $\vDash_{\mathtt{S4}^{\Delta}_{\mathcal{A}}} F$ or $\vDash_{\mathcal{A}} F$ for simplicity, if it is valid in all $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-models. The theory of the base, $Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$, is the set of all $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid formulas.

We need more terminology. Given awareness bases $\mathcal{A} = \langle \mathbf{A}, f\rangle$ and $\mathcal{B} = \langle \mathbf{B}, g\rangle$, $\mathcal{B} \subseteq \mathcal{A}$ if $\mathbf{B} \subseteq \mathbf{A}$ and $f(B) = g(B)$ for any $B \in \mathbf{B}$, and $\mathcal{A} \preccurlyeq \mathcal{B}$ if (1) $\mathcal{A} \subseteq \mathcal{B}$ and (2) $\mathbf{B} \subseteq Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$. For instance, for any $\mathcal{B} = \langle \mathbf{B}, g\rangle$ with $\mathbf{B} \subseteq Th(\mathtt{S4}^{\Delta}_{\emptyset})$, $\emptyset \preccurlyeq \mathcal{B}$, where $\emptyset$ is the empty base. $\preccurlyeq$ is not transitive.

Lemma 2. *For any awareness bases $\mathcal{B} \subseteq \mathcal{A}$, $Th(\mathtt{S4}^{\Delta}_{\mathcal{B}}) \subseteq Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$.*

Hence for any awareness base $\mathcal{A}$, $Th(\mathtt{S4}^{\Delta}_{\emptyset}) \subseteq Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$.

Our definition of semantics is very general. For any awareness base $\mathcal{A}$, even if it might contain formulas contradictory to each other, $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-models are defined. However, we are interested in the bases which contain only valid formulas, and can be used to characterize the logical strength of knowers by the number of logical truths they have. We need a constructive way to provide a general definition of bases of this kind.

Definition 4. *We say an awareness base $\mathcal{A} = \langle \mathbf{A}, f\rangle$ is an $\mathtt{S4}^{\Delta}$-awareness base if it satisfies one of the following three conditions:*

1. *$\mathcal{A} = \emptyset$.*
2. *There are awareness bases $\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$ such that $\emptyset = \mathcal{A}_0 \preccurlyeq \mathcal{A}_1 \preccurlyeq \cdots \preccurlyeq \mathcal{A}_n = \mathcal{A}$.*
3. *there is a collection of awareness bases $\{\mathcal{A}_i\}_{i\in\mathbb{N}}$ with $\mathcal{A}_i = \langle \mathbf{A}_i f_i\rangle$, such that $\emptyset = \mathcal{A}_0 \preccurlyeq \mathcal{A}_1 \preccurlyeq \mathcal{A}_2 \preccurlyeq \cdots$, and $\mathcal{A} = \bigcup \mathcal{A}_i$, that is, $\mathbf{A} = \bigcup \mathbf{A}_i$ and for any $A \in \mathbf{A}_i$, $f(A) = f_i(A)$.*

Then it is not difficult to check that if $\mathcal{A}$ is an $\mathtt{S4}^{\Delta}$-awareness base, for every $A \in \mathcal{A}$, $\vDash_{\mathcal{A}} A$.

We say an awareness base is finite if its base set is finite.

Lemma 3. *Given an $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{A}$ and a formula $F \in L_{\Delta}$, $\vDash_{\mathcal{A}} F$ if and only if there is a finite $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{B} \subseteq \mathcal{A}$, $\vDash_{\mathcal{B}} F$.*

This lemma can be shown by first proving the compactness theorem of our semantics, or is an immediate corollary of the completeness result that we will give later when axiom systems of our semantics are introduced. There is an interesting result about the critical awareness function.

Lemma 4. *For each $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{A}$, if $\alpha^{*}_{\mathcal{A}}(F)\downarrow$ then $\vDash_{\mathcal{A}} F$.*

The statement is not true for an arbitrary $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-awareness function. Before closing this subsection, we consider several conditions on collections of awareness functions which related to positive introspection, in contrast with conditions we will introduce later related to the negative introspection.

Definition 5. *We say a collection of awareness functions satisfies the **inner positive introspection** condition if every element of the collection satisfies the condition.*

Definition 6. *Given a model $M = \langle W, R, \{\alpha_w\}, \mathcal{V}\rangle$, we say the collection $\{\alpha_w\}$ satisfies the **outer positive introspection** condition if for any w with $(M, w) \Vdash \mathbf{K}F^i$, $\alpha_w(\mathbf{K}F^i) \leq i + 1$.*

This condition says that if at some world the knower knows some formula at some time, then at the world he is aware of the formula one time unit latter. Though, this condition is derivable.

Fact. *If a collection of awareness function satisfies the inner positive introspection condition, then the collection satisfies the outer positive introspection condition for any model the collection belongs to.*

Now we put all the conditions relative to the positive introspection together.

Definition 7. *Given a model $M = \langle W, R, \{\alpha_w\}, \mathcal{V}\rangle$, we say the collection $\{\alpha_w\}$ is **positive regular** if it is a monotonic and satisfies both inner and outer positive introspection conditions.*

2.4 More on $\mathtt{S4}^{\Delta}$-Awareness Bases

In our semantics, knowers with different awareness bases are regarded as having different epistemic strength, and modeled differently. The smallest awareness base is the empty base. We also can have a maximal awareness base. An $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{A} = \langle \mathbf{A}, f\rangle$ is maximal if $Th(\mathtt{S4}^{\Delta}_{\mathcal{A}}) \subseteq \mathbf{A}$. Let $\{\mathcal{A}_i\}_{i\in\mathbb{N}}$ be a collection of awareness bases with $\mathcal{A}_0$ the empty base and $\mathcal{A}_{i+1} = \langle \mathbf{A}_{i+1}, f_{i+1}\rangle$ such that $\mathbf{A}_{i+1} = Th(\mathtt{S4}^{\Delta}_{\mathcal{A}_i})$, and $f_{i+1}(A) = f_i(A)$ for any $A \in \mathbf{A}_i$, and let $\mathcal{A} = \bigcup \mathcal{A}_i$. Then by using Lemma 3, it can be proved that $\mathcal{A}$ is a maximal $\mathtt{S4}^{\Delta}$-awareness base.

Notice that there is more than one maximal awareness base. Different base functions will give us different maximal awareness bases. Call an awareness base with the constant function 0 principal, and let $\mathcal{A}$ be the principal maximal awareness base. Then if F is a $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid formula, so is $\mathbf{K}F^0$.

Obviously, knowers with maximal awareness bases are not realistic. Several other types of awareness bases have more intuitive appealing. Here's the list. Let $\mathcal{A}$ be an $\mathtt{S4}^{\Delta}$-awareness base.

1. There is a maximal base $\mathcal{B}$ such that $Th(\mathtt{S4}^{\Delta}_{\mathcal{B}}) \subseteq Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$.
2. For any $F \in Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$, $\alpha^*_{\mathcal{A}}(F)\downarrow$.
3. For any $F \in Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$, $\{\alpha(F)\}$ is bounded.
4. For any $F \in Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$, $\mathbf{K}F^i \in Th(\mathtt{S4}^{\Delta}_{\mathcal{A}})$ for some i.

The four definitions of these properties are from four different concerns. The first one concerns the relationship between the theory of the base and the theory of a maximal one. The statement in the second item is the converse of of Lemma 4, concerning critical awareness function. The third one concerns the awareness functions in general and the last one is about the theory of the base itself. Now what really interests us is that these awareness bases are all equivalent.

Theorem 1. *All the four properties of an* $\mathtt{S4}^{\Delta}$*-awareness base* $\mathcal{A}$ *defined above are equivalent.*

Definition 8. *An awareness base with one of the above property is called* full.

These full awareness bases turn out to play a special role for the realization theorem. In the next section we will see a concrete example of a full base.

3 Axiom Systems

Given an $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{A} = \langle \mathbf{A}, f \rangle$, the following is the axiom system $\mathtt{S4}^{\Delta}_{\mathcal{A}}$.

Definition 9.
Axioms
A0 classical propositional axiom schemes
A1 $\mathbf{K}(F \to G)^i \to \mathbf{K}F^j \to \mathbf{K}G^k \quad i, j < k$
A2 $\mathbf{K}A^i \to \mathbf{K}(\mathbf{K}A^i)^j \quad i < j$ *if* $A \in \mathbf{A}$ *and* $f(A) \leq i$
A3 $\mathbf{K}F^i \to \mathbf{K}F^j \quad i < j$
A4 $\mathbf{K}F^i \to \mathbf{K}(\mathbf{K}F^i)^j \quad i < j$
A5 $\mathbf{K}F^i \to F$
Inference Rule
R1 $\vdash G$, *if* $\vdash F \to G$ *and* $\vdash F$ *"modus ponens"*
R2 $\vdash \mathbf{K}A^i$ *if* $A \in \mathbf{A}$ *and* $f(A) \leq i$ *"$\mathcal{A}$-necessitation"*

We use $\vdash_{\mathtt{S4}^{\Delta}_{\mathcal{A}}} F$, or $\vdash_{\mathcal{A}} F$ for simplicity, to denote that F is a theorem in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$.

Theorem 2. *For any* $\mathtt{S4}^{\Delta}$*-awareness base* $\mathcal{A}$, $\vDash_{\mathcal{A}} F$ *if and only if* $\vdash_{\mathcal{A}} F$.

The proof is given in the appendix.

When our $\mathcal{A}$ is empty, the A2 axiom and the R2 rule are void. When $\mathcal{A}$ is maximal, the clause "$A \in \mathcal{A}$" can be replaced by "$\vdash A$", and when the base is principal, "and $f(A) \leq i$" can be removed. The most interesting awareness base will be the one containing all axioms. This one will be full.

Lemma 5. *Given an awareness base* $\mathcal{A} = \langle \mathbf{A}, f \rangle$, *if* $\mathbf{A}$ *contains all axiom instances of the system,* $\mathcal{A}$ *is full.*

Proof. With the completeness and soundness results above, it is sufficient to prove that if $\vdash_{\mathcal{A}} F$, then $\vdash_{\mathcal{A}} \mathbf{K}F^i$ for some i (hence if $\vDash_{\mathcal{A}} F$, $\vDash_{\mathcal{A}} \mathbf{K}F^i$). We prove the statement by induction on the length of the proof of F. Suppose F is an axiom, then by $\mathcal{A}$-necessitation, $\vdash_{\mathcal{A}} \mathbf{K}F^i$ for $i \geq f(A)$. If G is derived from $F \to G$ and F, by induction hypothesis $\vdash_{\mathcal{A}} \mathbf{K}(F \to G)^i$ and $\vdash_{\mathcal{A}} \mathbf{K}F^j$. Then using axiom A1, we have $\vdash_{\mathcal{A}} \mathbf{K}G^k$ for $k > i, j$. If $\mathbf{K}F^i$ is derived by $\mathcal{A}$-necessitation, by applying the axiom A2, $\vdash_{\mathcal{A}} \mathbf{K}(\mathbf{K}F^i)^j$ for $j > i$[3].

[3] This result is similar to Artemov's Internalization Theorem in LP [4].

Definition 10. *We say an awareness base $\mathcal{A}$ is* axiomatically appropriate *if its base set includes all axiom instances of the schemes in* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$.

When $\mathcal{A}$ is axiomatically appropriate, the clause "$A \in \mathcal{A}$" can be replaced by "A is an axiom" in the axiom system.

Notice that the A1 and A2 axioms are needed in the proof of Lemma 5. In our $\mathtt{S4}^{\Delta}$ axiom systems, all axioms except for the A2 axioms are $\mathtt{S4}^{\Delta}_{\emptyset}$-valid, and since all the functions of the A2 axioms can be replaced by the A4 axioms, thus an axiomatically appropriate base without the A2 axioms is still full. This makes it possible to have a full $\mathtt{S4}^{\Delta}$-awareness base $\mathcal{A}$ with $\emptyset \preccurlyeq \mathcal{A}$. But on the other hand, if we consider the Δ-counterpart of other normal modal logics, especially those without the A4 axioms, then the A2 axioms, which is corresponding to the 2b. condition of Definition 11, cannot be excluded from axiom appropriate bases to have full awareness bases.

4 Realization Theorem and Relations to LP

As mentioned earlier, our attitude to the problem of logical omniscience is that the standard epistemic logics, such as S4, do not model a knower who knows too much, but, instead, lack the expressive power such that when it is applied, we have to ascribe all logical truths as knowledge to the knower. To justify our claim, relations between S4 and $\mathtt{S4}^{\Delta}$ logics should be built. There are two directions of the relations. One is trivial. Roughly speaking, if we drop all the number labels from an $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ proof, we will have an S4 proof. This implies that every $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ theorem is an S4 theorem with time labels. For the other direction, in [17] it has been constructively proved that, for the principal axiomatically appropriate awareness base $\mathcal{A}$, every theorem in S4 can be translated to a theorem in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ by adding suitable time labels to modal formulas. This theorem is called the *realization theorem*. A sketch of the proof of this theorem is given at the end of this paper. The semantic proof of this theorem is in progress. From the experience of working on LP, the fullness of an awareness base would be sufficient for the realization theorem to hold. Now we can conclude that theorems in S4 provides us the relations between the knower's knowledges without considering the relations between the time points indicating when the knowledges is known, which are hidden in the S4 theorems and revealed in the theorems' Δ-counterparts in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$.

LP now is one of the family of *Justification Logics*. It is not only a logic passing the logical omniscience test, as we mentioned before, but also a logic with the justificatory complexity of knowledge acquisition is recorded. It has formula atoms like $t:F$ to mean t is a justification of F with t being an justification object. A efficient translation between proofs in $\mathtt{S4}^{\Delta}$ and proofs in LP has been given in [17]. The translation reflects an informal relations between justifications and time. Justifications need time, and we gain knowledge through time by giving justifications. LP has a more refined framework than $\mathtt{S4}^{\Delta}$. With justifications explicitly expressed, we can not only identify the temporal order of what we know but also trace the reasoning history of our knowledge. However, reasoning

in $\mathtt{S4}^{\Delta}$ is more intuitive and working on natural numbers with their linearity is easier than directly dealing with justification objects.

5 Variations

Our logic $\mathtt{S4}^{\Delta}$ is adapted from S4, both semantically and syntactically. We add an $\mathtt{S4}^{\Delta}$-awareness function to each world in a S4 model to get the $\mathtt{S4}^{\Delta}$-semantics and add number labels to axiom schemes to have Δ-style axiom systems. By similar manner we can have Δ-counterparts of the neighboring logics of S4, semantically and syntactically. Let $M = \langle W, R, \{\alpha_w\}, \mathcal{V}\rangle$ be a Δ-model. In this section, awareness functions are not supposed to satisfy the inner positive introspection condition. The following is a table of the conditions on frames and collections of awareness functions of models that the complete and sound semantics of these sublogics of $\mathtt{S4}^{\Delta}$ should satisfy.

	$\langle W, R\rangle$	$\{\alpha_w\}$
$\mathtt{K}^{\Delta}$	no condition	no condition
$\mathtt{KT}^{\Delta}$	reflexive	no condition
$\mathtt{K4}^{\Delta}$	transitive	positive regular
$\mathtt{KT4}^{\Delta}$	transitive and reflexive	positive regular

For these Δ-logics, when an awareness base is axiomatically appropriate with respect to their axiom systems, the base is full. Here the axiom A2 plays his role.

Now add the 5 axiom[4]: "$\neg\mathbf{K}F^i \rightarrow \mathbf{K}\neg\mathbf{K}F^j$ for $i < j$." into the picture. We say an awareness function α satisfies the **inner negative introspection** condition if $\alpha(\neg\mathbf{K}F^i) \leq i+1$, provided $\alpha(F) \not\leq i$, and a collection of awareness functions satisfies the condition if every element of the collection satisfies the condition. Given a model $M = \langle W, R, \{\alpha_w\}, \mathcal{V}\rangle$, we say the collection $\{\alpha_w\}$ is **anti-monotonic** if for any $w\mathcal{R}w'$, whenever $\alpha_w(F) \not\leq i$, then $\alpha_{w'}(F) \not\leq i$, and we say the collection $\{\alpha_w\}$ satisfies the **outer negative introspection** condition if $\alpha_w(\neg\mathbf{K}F^i) \leq i+1$, provided $(M, w) \Vdash \neg\mathbf{K}F^i$. Finally, we call the collection is **negative regular** if the collection is anti-monotonic and satisfies both the inner and outer negative introspection conditions. Then for those systems with the 5 axiom, in their complete and sound models, the frame should be *Euclidean*, and the collection of awareness functions is negative regular.

Fact. *If a collection of awareness function in a model satisfies the outer negative introspection condition, then the collection satisfies the inner negative introspection condition.*

Acknowledgements. I would like to thank profefossor S. Artemov for encouraging this work. And thank an anonymous reviewer for many practical suggestions to improve this paper.

[4] This work here is adapted from [20] and [21], more related to [20], but different.

References

1. Hintikka, J.: Knowledge and Belief: An Introduction to the Logic of the Two Notions. Cornell University Press (1962)
2. Artemov, S.N.: The logic of justification. The Review of Symbolic Logic 1(4), 477–513 (2008)
3. Artemov, S.N.: Operational modal logic. Technical Report MSI 95-29, Cornell University (1995)
4. Artemov, S.N.: Explicit provability and constructive semanticsexplicit provability and constructive semantics. Bulletin of Symbolic logic 7(1), 1–36 (2001)
5. Rantala, V.: Impossible worlds semantics and logical omniscience. Acta Philosophica Fennica 35, 107–115 (1982)
6. Fagin, R., Halpern, J.: Belief, awareness and limited reasoning. Artificial Intelligence 34, 39–76 (1988)
7. Konolige, K.: A Deduction Model of Belief. Morgan Kaufmann, San Francisco (1986)
8. Moreno, A.: Avoiding logical omniscience and perfect reasoning: A survey. AI Communications 11(2), 101–122 (1998)
9. Fagin, R., Halpern, J., Moses, Y., Vardi, M.: Reasoning About Knowldge. MIT Press, Cambridge (1995)
10. 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)
11. Asher, N.: Reasoning about belief and knowledge with self-reference and time. In: LFCS, pp. 61–81. Morgan Kaufmann, San Francisco (1988)
12. Kraus, S., Lehmann, D.: Knowledge, belief and time. Theoretical Computer Science 58, 155–174 (1988)
13. Ladner, R.E., Reif, J.H.: The logic of distributed protocols. In: Halpern, J.Y. (ed.) Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference, pp. 207–222. Morgan Kaufmann, San Francisco (1986)
14. Lehmann, D.: Knowledge, common knowledge, and related puzzles. In: Proceedings of 3rd ACM Symp. on Principles of Distributed Computing, pp. 62–67 (1984)
15. Parikh, R., Ramanujam, R.: Distributed processes and the logic of knowledge. In: Proceedings of the Conference on Logic of Programs, London, UK, pp. 256–268. Springer, Heidelberg (1985)
16. Sato, M.: A study of kripke-type models for some modal logics by gentzen's sequential method. In: Publications of the Research Institute for Mathematical Sciences, vol. 13, p. 381. Kyoto University (1977)
17. Wang, R.J.: On proof realization on moda logic. Technical Report TR-2009003, CUNY PhD Program in Computer Science (2009)
18. Fitting, M.C.: The logic of proofs, semantically. Annals of Pure and Applied Logic 132, 1–25 (2005)
19. Fitting, M.C.: A logic of explicit knowledge. In: Behounek, L., Bilkova, M. (eds.) Logica Yearbook 2004, Filosophia, pp. 11–22 (2005)
20. Pacuit, E.: A note on some explicit modal logics. In: Proceedings of the Fifth Panhellenic Logic Symposium, pp. 117–125 (2005)
21. Rubtsova, N.: Evidence reconstruction of epistemic modal logic S5. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, pp. 313–321. Springer, Heidelberg (2006)

Appendix

Some Proofs

Proof of Lemma 1

Proof. Let $a = \min\{j \mid f(A) = j, \text{ for every } A \in \mathbf{A}\}$ with f the base function. The construction is straightforward. First, let $\alpha^*_{\mathcal{A}}(F) = a$ for any $f(F) = a$. Then suppose the construction of $\alpha^*_{\mathcal{A}}$ have been completed up to n, we just define $\alpha^*(G) = n + 1$ for whatever formula G needed to be defined. That is if $\alpha^*_{\mathcal{A}}(G)$ is not defined yet when, say, $\alpha^*_{\mathcal{A}}(F \to G)$ and $\alpha^*_{\mathcal{A}}(F)$ is defined, or $G \in \mathbf{A}$ with $f(G) = n + 1$, define $\alpha^*_{\mathcal{A}}(G) = n + 1$. Continue this process, we have the unique critical awareness function.

Proof of Theorem 1

Proof. We prove the directions from 1 to 4 and 2 to 1, The directions form 4 to 3 and 3 to 2 are trivial. Suppose $\models_{\mathcal{A}} F$ and there is a maximal base $\mathcal{A}'$ such that $\models_{\mathcal{A}'} F$, then $F \in \mathcal{A}'$. Since $\models_{\mathcal{A}'} \mathbf{K}F^i$ for some i, $\models_{\mathcal{A}} \mathbf{K}F^i$. This proves the direction from 1 to 4.

Now we prove the direction from 2 to 1. Suppose for every $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid formula F, $\alpha^*_{\mathcal{A}}(F)$ is defined. We define $\mathcal{A}' = \langle \mathbf{A}', f' \rangle$ with $\mathbf{A}' = \{F \mid \models_{\mathcal{A}} F\}$ and $f'(F) = \alpha^*_{\mathcal{A}}(F)$. Since for any $F \in \mathbf{A}'$ and any $\mathcal{A}$-awareness function α, $\alpha(F) \leq f'(F) (= \alpha^*_{\mathcal{A}}(F))$, α is an $\mathcal{A}'$-awareness function. Hence every $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-model is a $\mathtt{S4}^{\Delta}_{\mathcal{A}'}$-model and every $\mathtt{S4}^{\Delta}_{\mathcal{A}'}$-valid formula is an $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid formula. By the definition of $\mathcal{A}'$, every $\mathtt{S4}^{\Delta}_{\mathcal{A}}$-valid formula is in $\mathcal{A}'$. $\mathcal{A}'$ is maximal.

Proof of Theorem 2

Proof. The soundness part is straightforward. For the completeness part, we construct a model composed of maximal $\mathcal{A}$-consistent sets. A set S of formulas in L_{Δ} is said to be $\mathcal{A}$-consistent if there is no finite subset $\{F_1, \ldots, F_n\}$ of S such that $\vdash_{\mathcal{A}} \neg(F_1 \wedge \ldots \wedge F_n)$. The construction of a maximal such set is by the standard Lindenbaum construction. Let W be the set of all maximal $\mathcal{A}$-consistent sets, and for any Γ, $\Gamma' \in W$, $\Gamma R \Gamma'$ if and only if $\Gamma^\sharp \subseteq \Gamma'$ where $\Gamma^\sharp = \{F \mid \mathbf{K}F^i \in \Gamma\}$. $\alpha_\Gamma(F) = \min\{i \mid \mathbf{K}F^i \in \Gamma\}$ and $\mathcal{V}(P) = \{\Gamma \mid P \in \Gamma\}$. We claim this $M = \langle W, R, \{\alpha_w\}, \mathcal{V} \rangle$ is a $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ model. R is transitive and reflexive because of the axioms A2 and A4. α_Γ satisfy the initial condition due to the $\mathcal{A}$-necessitation rule, and it is not difficult to check that α_Γ satisfies all other conditions by applying these conditions' corresponding axioms. The collection of these awareness functions also satisfies the monotonicity condition. When $\alpha_\Gamma(F) = i$, $\mathbf{K}F^i \in \Gamma$. Since $\models_{\mathcal{A}} \mathbf{K}F^i \to \mathbf{K}(\mathbf{K}F^i)^j$, $\mathbf{K}(\mathbf{K}F^i)^j \in \Gamma$, so $\mathbf{K}F^i \in \Gamma'$ for any $\Gamma^\sharp \subseteq \Gamma'$. $\alpha_{\Gamma'}(F) \leq i$.

Then is the truth lemma, that is, for every Γ, $F \in \Gamma$ if and only if $(M, \Gamma) \Vdash F$. The proof is by inductin on the complexity of formulas. Most of cases are trivial. We prove the modal case. If $(M, \Gamma) \Vdash \mathbf{K}F^i$ then $\alpha_\Gamma(F) \leq i$, $\mathbf{K}F^i \in \Gamma$. For the other direction, if $\mathbf{K}F^i \in \Gamma$, $\alpha_\Gamma(F) \leq i$ and for any $\Gamma^\sharp \subseteq \Gamma'$, $F \in \Gamma'$, so by induction $(M, \Gamma') \Vdash F$. Hence $(M, \Gamma) \Vdash \mathbf{K}F^i$. Now suppose F is not provable in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$, $\neg F$ is $\mathcal{A}$-consistent. $(M, \Gamma) \nVdash F$ with Γ a maximal $\mathcal{A}$-consistent set containing $\neg F$.

The Realization Theorem

The realization procedure in [17] is the following. A Δ-style cut-free Gentzen system $\mathtt{S4}^{\Delta}G^{-}$ corresponding to a cut-free Gentzen system $\mathtt{S4}G^{-}$ of $\mathtt{S4}$ is introduced. Every rule in the system is derivable in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ with $\mathcal{A}$ the principle maximal awareness base. Then it is shown that every proof in $\mathtt{S4}G^{-}$ can be turned into a proof in $\mathtt{S4}^{\Delta}G^{-}$ by supplementing suitable natural number labels to every modal formulas. So it follows that every $\mathtt{S4}$ theorem can be converted to an $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ theorem.

A sequent $\Gamma \Rightarrow \Gamma'$ is a pair of finite multisets Γ, Γ' of formulas. It is convenient to view a sequenct as a formula $C_1 \rightarrow (\ldots \rightarrow (C_n \rightarrow \bigvee \Gamma')\ldots)$ here. Given a multiset $\Gamma = \{C_i\}$ of formulas in $L_{\mathbf{K}}$, $\mathbf{K}\Gamma = \{\mathbf{K}C_i\}$. Given a multiset $\Gamma = \{C_i\}$ of formulas in L_{Δ}, $\mathbf{K}\Gamma^{\iota} = \{\mathbf{K}C_i^{j_i}\}$, for j_i a number in the multiset ι. $|\Gamma|$ is the number of formulas in Γ. Here's the Gentzen systems.

Definition 11 ($\mathtt{S4}G^{-}$)**.**
The only axiom is that $P \Rightarrow P$, for a propositional letter P.
The rules for weakening (W) and constraction (C)

$$LW\,\frac{\Gamma \Rightarrow \Gamma'}{A, \Gamma \Rightarrow \Gamma'} \quad RW\,\frac{\Gamma \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma', A}$$

$$LC\,\frac{A, A, \Gamma \Rightarrow \Gamma'}{A, \Gamma \Rightarrow \Gamma'} \quad RC\,\frac{\Gamma \Rightarrow \Gamma', A, A}{\Gamma \Rightarrow \Gamma', A}$$

The classical logical rules

$$L\neg\,\frac{\Gamma \Rightarrow \Gamma', A}{\neg A, \Gamma \Rightarrow \Gamma'} \quad R\neg\,\frac{\Gamma, A \Rightarrow \Gamma'}{\Gamma \Rightarrow \Gamma', \neg A}$$

$$L\rightarrow\,\frac{\Gamma \Rightarrow \Gamma', A \quad B, \Gamma \Rightarrow \Gamma'}{A \rightarrow B, \Gamma \Rightarrow \Gamma'} \quad R\rightarrow\,\frac{A, \Gamma \Rightarrow \Gamma', B}{\Gamma \Rightarrow \Gamma', A \rightarrow B}$$

The modal rules are

$$R\mathbf{K}\,\frac{A, \Gamma \Rightarrow \Gamma'}{\mathbf{K}A, \Gamma \Rightarrow \Gamma'} \quad R\mathbf{K}\,\frac{\mathbf{K}\Gamma \Rightarrow A}{\mathbf{K}\Gamma \Rightarrow \mathbf{K}A}$$

Definition 12 ($\mathtt{S4}^{\Delta}G^{-}$)**.**
The language for $\mathtt{S4}^{\Delta}G^{-}$ is L_{Δ}. It is the $\mathtt{S4}G^{-}$ with the following Δ-modal rules

$$L\mathbf{K}\,\frac{A, \Gamma \Rightarrow \Gamma'}{\mathbf{K}A^{i}, \Gamma \Rightarrow \Gamma'}\,,\ \textit{for any } i$$

$$R\mathbf{K}\,\frac{\mathbf{K}\Gamma^{\iota} \Rightarrow A}{\mathbf{K}\Gamma^{\iota} \Rightarrow \mathbf{K}A^{i}}\,,\ \textit{for any } i > \max(\iota) + |\Gamma| + 1,\ \textit{when } |\Gamma| \neq 0,\ \textit{and for any } i \textit{ when } |\Gamma| = 0$$

Lemma 6. *Every rule in the system is derivable in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ with $\mathcal{A}$ the principle maximal awareness base.*

Proof. It is sufficient to check the Δ-modal cases. The case for the left modal rule is also trivial. By induction, it can be checked that for $|\Gamma| > 0$ and $i > \max(\iota, e) + |\Gamma| + 1$, $\mathbf{K}(\mathbf{K}\Gamma^{\iota} \Rightarrow A)^{e} \rightarrow (\mathbf{K}\Gamma^{\iota} \Rightarrow \mathbf{K}A^{i})$ is provable in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ with $\mathcal{A}$ the principle maximal awareness base. Then it follows that the right modal rule is derivable.

Lemma 7. *Every* $\mathtt{S4}G^-$ *proof is a proof of* $\mathtt{S4}^{\Delta}G^-$ *without number labels.*

Proof. Call the formulas with the modal operator **K** as the main connectives as m-formulas. Let all negative m-formula occurrences have label 0, and all positive m-formula occurrences have the label equal to the nubmer of formula occurrences in the $\mathtt{S4}G^-$ proof. Then the condition for the right Δ-modal rule will be satisfied. So the resulting sequent tree is a proof in $\mathtt{S4}^{\Delta}G^-$.

In [17], a procedure coverting a proof in $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ with $\mathcal{A}$ the principal maximal awareness base to a proof in $\mathtt{S4}^{\Delta}_{\mathcal{A}'}$ with $\mathcal{A}'$ the principal axiomatically appropriate awareness base is also given . The statement is jsutified.

Theorem 3. *Given* $\mathcal{A}$ *the principal axiomatically appropriate awareness base, every* S4 *theorem is a* $\mathtt{S4}^{\Delta}_{\mathcal{A}}$ *theorem without number labels.*

Author Index